Design and Implementation of Text-Based Information Retrieval System using Fuzzy Logic

A thesis
Submitted to the College of Science Baghdad University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

By
Ibrahim A. Hameed
(B.Sc. 2008)

Supervisor
Dr. Loay E. George

September 2011                      Shawal 1423
I certify that this thesis was prepared under my supervision at the Department of Computer Science/College of Science/Baghdad University, by **Ibrahem Amer Hameed** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

**Supervisor**

Signature:

Name: **Dr. Loay E. George**

Title: Assistant Professor

Date: / /2011

**The Head of the Department Certification**

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:

Name: **Dr. Loay E. George**

Title: Head of the Department of Computer Science, Baghdad University

Date: / /2011
Acknowledgment

I express a special thank to Dr. Loay E. George, beside his valuable guidance and ideas to complete this work, I thank him for the interesting discussions that we had, which included various issues and topics of life. For those hours and ideas that you shared with me, I show my sincere thanks and gratitude.

Finally, to everyone who helped me and supported me, I say "thank you".
ABSTRACT

Since Web is a rich source of information, it becomes necessary to invent assistance systems to efficiently and automatically search, and nominate for more suitable and convenient Web page documents. This kind of tools will help the user to reach its needs on the Web media without wasting time and effort. The purpose of this study is to construct an automated information retrieval system that helps to retrieve the HTML documents that have a textual content similar to any document chosen by a user.

The textual content in every electronic repository (including the Web) is statistically variable and has complex behavior. It is noticed that the classical criteria for similarity don’t give an encouraging results in terms of accuracy and rationality metrics. So, in the proposed system a logical paradigm is developed which fundamentally depends on the fuzzy logic criteria to soften the decisions of the matching between the textual contents and makes the results more rational and more reasonable for the user.

The proposed system composed of two phases the enrollment phase and the retrieval phase. In the enrollment the extracted keywords from documents are stored along with their textual features, the Web documents are preprocessed using a sequence of text mining operations and then a useful knowledge (i.e., the set of keywords) is extracted and deposited in a dedicated database. At the retrieval phase, when the client issues a query document request the system matches this query document with every document stored in the database (after formulating the query document to a format compatible with the indexed feature vectors). As a matching result is a score value is given which depends on the number of common keywords found in both matched documents and on the degree of relative significance of these common keywords in both documents. Each matching instance is fuzzyfied using an s-shape membership function. Then by applying the
criteria "the highest the match score the best the match" (due to matching the query with all documents pre-deposited in the database) is sorted in descending order, in order to nominate the top listed documents as a query results.

The results of the conducted performance tests showed that the usage of fuzzy logic soften the matching decisions and lead to better results than using the traditional hard computing techniques. The best found precision and recall values were 0.99 and 0.66, respectively.
# TABLE OF CONTENTS

## Chapter one: General Introduction

1.1 Overview ............................................................................................................. 1  
1.2 Brief History ....................................................................................................... 2  
1.3 Data Mining ......................................................................................................... 3  
1.4 Challenges of Mining the World Wide Web ....................................................... 5  
1.5 Literature Survey ................................................................................................. 6  
1.6 Aim of the Thesis ................................................................................................. 7  
1.7 Thesis Outline ...................................................................................................... 8-9

## Chapter 2: Theoretical Background

2.1 Introduction ......................................................................................................... 10  
2.2 Knowledge vs. information ............................................................................... 10  
2.3 Information Retrieval ......................................................................................... 11  
   2.3.1 Machine Learning for Information Retrieval .............................................. 12  
   2.3.2 Web Mining ............................................................................................... 13  
   2.3.3 Retrieval vs. Mining ............................................................................... 14  
   2.3.4 Web Data Types ..................................................................................... 14  
   2.3.5 Web Mining Applications ...................................................................... 15  
2.4 Stemming ............................................................................................................. 24  
   2.4.1 Morphology (Linguistics) Overview ......................................................... 25  
   2.4.2 Krovetz Stemming Algorithm ................................................................... 25  
2.5 Fuzzy Logic ......................................................................................................... 26  
   2.5.1 Fuzzy Set Representation ........................................................................ 26
Chapter 3: TBIR system

3.1 Introduction ..................................................................................... 48
3.2 The Architecture of TBIR System .................................................. 48
3.3 Client and Server Architecture ....................................................... 49
3.4 Enrollment phase ............................................................................ 50
3.5 Retrieval Phase ................................................................................ 71
3.6 Search technique ............................................................................. 77

Chapter 4: Test Results

4.1 Introduction ..................................................................................... 81
4.2 Test Strategy .................................................................................... 81
4.3 Test Material .................................................................................... 82
4.4 Effect of the Bonus Increment Parameter (α) .................................. 83
4.5 Effect of the Matching Score Weight Factor (C) ............................ 84
4.6 Effect of the Database Document Record Size (DB\textsubscript{recsize})............ 85
4.7 Effect of Query Document Vector Window Size (Q\textsubscript{recsize})............ 86
4.8 Effect of S\_ Function Exponent Factor (R)........................................... 87
4.9 Text Operations Effect ..................................................................... 88
  4.9.1 Stemming/Stop Word Removal Dependency ................................ 88
  4.9.2 Text Operations Effect on Retrieval ............................................ 92
4.10 Nonlinear Function Effect on the Retrieval ........................................ 89
4.10 Frequency Based vs. Exclusive search ........................................... 91-92

Chapter 5: Conclusions and Future Work

5.1 Conclusions .................................................................................... 93
5.1 Suggestions ..................................................................................... 94

References
Appendix A
Appendix B
LIST OF TABLES

2.1 Differences between IR and IE .........................................................10
2.2 Contingency performance parameters ...........................................19
2.3 Bit groupings ..............................................................................34
3.1 example of stemming types.............................................................58
3.2 The keyword's data table...............................................................62
3.3 The structure of database table.......................................................71
3.4 Bonus determination example ..........................................................76
4.1 The classes of subjects used for TBIR ............................................85
4.2 The test of the bonus increment parameter....................................86
4.3 Matching score weight factor............................................................87
4.4 Database document list size effect on the retrieval of TBIR.............88
4.5 Query document list size effect on the retrieval of TBIR...............89
4.6 The exponent of the nonlinear function test....................................90
4.7 Stop words removal and stemming relationship............................91
4.8 The effect of text operations (i.e., stemming and stop words removal) on the retrieval.................................................................92
4.9 The effect of s-mapping function....................................................93
4.9 The Frequency-based vs. Executive search .....................................95
LIST OF FIGURES

1.1 Architecture of a typical data mining system.................................................4
1.2 Number of Websites (1990-2008)......................................................................5
2.1 Web mining architecture....................................................................................14
2.2 Web usage mining...............................................................................................16
2.3 Classical information retrieval system...............................................................18
2.4 Similarity in VSM..............................................................................................24
2.5 An example of the membership grades of the fuzzy set..................................27
2.6 Trapezoidal Membership Function....................................................................28
2.7 Triangular membership function........................................................................29
2.8 S-shaped Membership Function.........................................................................30
2.9 Gaussian Membership Function..........................................................................30
2.10 Fuzzy operations...............................................................................................32
2.11 An example of hash function............................................................................33
2.12 Visual illustration of a hash table.................................................................33
2.13 Communication steps between Web server and Web browser....................33
2.14 Two tier architecture ..................................................................................37
2.15 Three tier architecture..................................................................................38
2.16 Characteristics of typical Web document .........................................................40
2.17 An illustrative example of an HTML file.....................................................42
2.18 JavaScript code..............................................................................................44
2.19 CSS Selector.................................................................................................45
2.20 Internal style sheet..........................................................................................46
3.1 TBIR system architecture................................................................................49
3.2 The tasks accomplished on both server and client sides..............................50
3.3 Enrollment process...........................................................................................51
3.4 Binary encoding for a set of textual features.................................70
3.5 Decoding using the "AND" bit-wise operation.................................70
3.6 S-Shaped MF ...........................................................................75
3.7 TBIR retrieval phase.................................................................78
3.9 Frequency-based search method..................................................81
4.1 The effect of text operation effect on the retrieval.........................93
## LIST OF ABRIVIATIONS AND ACRONYMS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>CERN</td>
<td>Centre European de Recherché Nucléaire</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascaded Style Sheet</td>
</tr>
<tr>
<td>DIK</td>
<td>Data, Information and Knowledge</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>DNS</td>
<td>Dynamic Name System</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>FCM</td>
<td>Fuzzy C-Means</td>
</tr>
<tr>
<td>FLS</td>
<td>Fuzzy Logic Systems</td>
</tr>
<tr>
<td>FL</td>
<td>Fuzzy Logic</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Mark-Up Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer protocol</td>
</tr>
<tr>
<td>IMC</td>
<td>Internet Mail Consortium</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ID3</td>
<td>Iterative Dichotomize 3</td>
</tr>
<tr>
<td>IR</td>
<td>Information Retrieval</td>
</tr>
<tr>
<td>IRS</td>
<td>Information Retrieval System</td>
</tr>
<tr>
<td>IETF</td>
<td>International Engineering Task Force</td>
</tr>
<tr>
<td>IANA</td>
<td>Internet Assigned Number Authority</td>
</tr>
<tr>
<td>KDD</td>
<td>Knowledge Discovery from Data base</td>
</tr>
<tr>
<td>KM</td>
<td>Knowledge Discovery</td>
</tr>
<tr>
<td>MIME</td>
<td>Multipurpose Internet Mail Extensions</td>
</tr>
<tr>
<td>NCSA</td>
<td>National Centre for Super Computing Applications</td>
</tr>
<tr>
<td>MF</td>
<td>Membership Function</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comments</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>SVG</td>
<td>Scalable Vector Graphics</td>
</tr>
<tr>
<td>SGML</td>
<td>Standard Generalized Mark-up Language</td>
</tr>
<tr>
<td>TF</td>
<td>Term Frequency</td>
</tr>
<tr>
<td>TERC</td>
<td>National Health Service</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>UTF-8</td>
<td>Unicode Transformation Format-8.</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Mark up Language</td>
</tr>
<tr>
<td>XUL</td>
<td>XML UI (User Interface) Language</td>
</tr>
<tr>
<td>XHTML</td>
<td>Extended HTML</td>
</tr>
</tbody>
</table>
Chapter One
General Introduction

1.1 Overview

The World Wide Web, or simply the Web, serves as a huge, widely distributed, global information service center for news, advertisements, consumer information, financial management, education, government, e-commerce, and many other information services [Wes05]. The Web is a popular and interactive medium to collect, disseminate, and access an increasingly huge amount of information.

Nowadays, information access on the Web is the main problem of the so-called Web Information Retrieval (IR). The Web represents a new framework that is rather different with respect to the traditional IR framework and sheds new and difficult challenges [Enr09]. Web presents particular characteristics that limit the existing IR technologies and determines the need to design new information access technologies [Chr09]:

- Web is possibly the biggest dynamic information resource.
- Web presents a structure of linked pages.
- Web is growing and updating at a very high rate.
- Web is very heterogeneous.

Imprecision and vagueness characterize several tasks in Web IR, such as assessing the relevance of Web pages, dealing with the multimedia nature of information, identifying spam problem, discovering deception, etc. Furthermore, due to this complexity, any major advance in the field of information access on the Web requires the application of intelligent techniques [Kai06].

Soft Computing (SC) techniques constitute a synergy of methodologies (e.g. fuzzy logic, neural networks, probabilistic reasoning,
rough-set theory, evolutionary computing and parts of machine learning theory) they are useful for solving problems requiring some form of intelligence [Chr09]. The basis of SC is its tolerance to imprecision, uncertainty, partial truth, and approximation. Because of these properties SC can provide very powerful tools for modeling the activities related with the Web information access problem [Enr09].

1.2 Brief History

The need to store and retrieve written information became increasingly important over centuries, especially with inventions like paper and the printing press. Soon after computers were invented, people realized that they could be used for storing and mechanically retrieving large amounts of information. Several key developments in the field happened in the 1960s. The most notable one was the development of a system for the mechanical analysis and retrieval of text called (SMART) by Gerard Salton and his students, first at Harvard University and later at Cornell University; this system allowed researchers to experiment their ideas and to improve search quality. The use of SMART system for experimentation coupled with good evaluation methodology had allowed for rapid progress in IR field, and paved the way for many critical developments [Chr95].

The 1970s and 1980s saw many developments built on the achieved advances of the 1960s. Various models for doing document retrieval were developed and advances were made along all dimensions of the retrieval process. These new models/techniques were proven experimentally, to be effective on small text collections. However, due to lack of availability of large text collections, the question whether these models and techniques would scale to larger corpora remained unanswered. This had been changed in 1992; with the inception of Text Retrieval Conference (TREC) [Dav96]. TREC is a series of evaluation conferences sponsored by various US Government agencies under the auspices of the National Institute of Standards and Technology (NIST), which aimed to encourage
research in information retrieval from large text collections. With large text collections available under TREC, many old techniques were modified, and many new techniques were developed and others still being developed. TREC has also branched IR into related but important fields like retrieval of spoken information, non-English language retrieval, information filtering, user interactions with a retrieval system, and so on. The algorithms developed in IR were the first ones to be employed for searching the World Wide Web from 1996 to 1998 [Ami99].

1.3 Data Mining

Data mining refers to extracting or “mining” knowledge from large amounts of data. Its synonym is Knowledge Discovery from Data (KDD) [Jia06]. Huge masses of data are daily generated from cash registers, from documents scanning, specific databases throughputs, and Web data. The extracted data may be textual, multimedia, structural, usage or Meta data. These mined data are explored, analyzed, reduced, and reused. A lot of search efforts were performed using different models; for example, they were proposed for predicting sales, marketing response, and profits. Classical statistical approaches are fundamental to data mining [Bor05]. Automated Artificial Intelligence (AI) methods are also being developed, and many of these methods are still under development. However, systematic exploration through classical statistical methods is still the basis of data mining [Dav08].

The essential stage in the process of knowledge discovery consists of an iterative sequence of the following steps [Two99]:

1. Data cleaning: to remove noise and inconsistent data.
2. Data integration: where multiple data sources may be combined.
3. Data selection: where data relevant to the analysis task are retrieved from the database.
4. Data transformation: where data are transformed or consolidated into forms appropriate for mining.
5. Data mining: is an essential process where intelligent methods are applied in order to extract data patterns.

6. Pattern evaluation: is to identify the truly interesting patterns representing knowledge based on some interestingness measures.

7. Knowledge presentation: where visualization and knowledge representation techniques are used to present the mined knowledge to users.

The architecture of a typical data mining system may have the following major components: (i) database, (ii) data warehouse, (iii) information repository (like, WWW). The third component, used as information container, is one or a set of databases, data warehouses, spreadsheets, or other kinds of information repositories. Data cleaning and data integration techniques may be performed on the data. Figure (1.1) illustrates the architecture of a typical data mining system [Jia06].

![Architecture of a typical data mining system](image)
Chapter One  General Introduction

The wild growth expansion of Internet has caused an exponential expansion in information sources and, also, in information-storage units. An illustrative example is given in Figure (1.2), it can be noticed that a dramatic increase of Internet hosts has started from 2000; these numbers are directly proportional with the amount of data stored on the Internet [Meh03].

![Number of Websites (1990-2008)](image)

Fig. (1.2) Number of Websites (1990-2008)

1.4 The Challenges of Mining the World Wide Web

The Web poses great challenges as an effective information resource for knowledge discovery. Among these challenges [Wes05] are:

1. The Web seems to be too huge for effective data warehousing and data mining. Now, the size of the Web is of order the of hundreds of terabytes and is still growing rapidly. Many organizations and societies in the world place most of their public-accessible information on the Web. It is barely possible to set up a data warehouse to replicate, store, or integrate all of the data on the Web.

2. The complexity of Web pages is far greater than that of any traditional text document collection. Web pages lack a unifying structure. They contain far more authoring style and content variations than any set of books or other traditional text-based documents.
3. The Web is considered as a huge digital library; however, a tremendous number of documents deposited in this library are not arranged according to any particular sorted order. It can be very challenging to search for the information you desire in such a library.

4. The Web is a highly dynamic information source whose information is constantly updating. News, stock, markets, weather, sports, shopping, company advertisements, linkage information and access records are updated frequently.

5. The Web serves a broad diversity of user communities. The Internet currently connects more than 100 million workstations, and its users community is still rapidly expanding. Users may have very different backgrounds, interests, and usage purposes.

6. Only a small portion of the information on the Web is truly relevant or useful for any user. It is true that a particular person is generally interested in only a tiny portion of the Web, while the rest of the Web contains information that is uninteresting to the user and may take place the desired search results.

These challenges promote research activities, to develop efficient and effective tools for discovery and use of resources on the Internet. The Web mining tools, which derive from the data mining field, are used to face these challenges.

1.5 Literature Survey

Among the large published research works in the field of the text information retrieval some of them are introduced below:

1. At 2005, Ihab proposed a Web search engine; the search process consists of crawling, indexing, and ranking (to all the Web pages). The indexing results are stored in a database. Any user can use the Web site search engine to find the information relevant to his query; this accomplished by searching the database and then returns a list of URL of to all Web pages contain information related to the user query. The
ranking part depends on the word attributes (such as font size, font style, font color, position of the word in the page, link text, title, and header). He introduced an indexing scheme called inverted index; it is created by sorting the indexes using the Improved Quick and Insertion sort methods. The ranking part is based on the word attributes, using the link structure of the Web [Moh05].

2. At 2006, Shaimaa, in her thesis, investigated the usage of Web mining techniques for online news sites. Her proposed method had been applied on dynamic Web sites to extract news reports. The most important objective of the system is the discovery of ephemeral associations that can be translated into knowledge about interest of a society and social behavior. The discovery of such kind of news trends helps to interpret the society interests, uncover hidden information about the relationships between the events in social life, and to measure the social importance of many events [Bah06].

3. At 2006, Saran [Cha06] gave in his thesis a description of a document classification system. His introduced system is based on utilizing both high-level features of HTML documents and the traditional term frequency information. He referred that the introduced system can achieve better classification results than systems that relay solely on text or document structure.

4. At 2007, Stephan accomplished a prototype of an information retrieval system that retrieves medical documents, the system named "Medical Information Retrieval System" MIRS. The techniques for indexing and the retrieval of relevant information in textual documents have been presented and an approach of Machine Learning has been adopted in classifying the document [Spa07].

5. At 2008, Huda suggested in her thesis a method to retrieve information using Web content mining technique. The introduced system takes into consideration the words attributes to calculate their weights. She used fuzzy logic to determine the similarity of documents [Tal08].
6. At 2010, Gupla et al. introduced a technique called “Retrieval of Web documents using a fuzzy hierarchical clustering”. The proposed technique creates clusters of web documents to speed up the retrieval process. The proposed technique utilizes fuzzy logic approach to improve the relevancy factor. This technique keeps the related documents in same cluster; so that searching of documents becomes more efficient in terms of time complexity [Dee10].

7. Kobayashi et al, at 2010, had made a review for the growth of the Internet and technologies that are useful for information search and retrieval on the Web. They listed some of notable remarks on the growth of the Internet and on the technologies useful for information search and retrieval on the Web [Mei10].

1.6 Aim of the Thesis

The aim of this thesis is to develop an automated and intelligent information retrieval system that retrieves the HTML files and ranks them according to the degree of their similarities with the query HTML document. The most similar documents are top ranked in the fetched list of addresses. The developed information retrieval system depends only on the textual content of Web documents.

Some of Fuzzy Logic based concepts is used to perform the matching and ranking of the document instead of using the traditional hard computing techniques. The frequency of occurrence of each extracted word, its printing attributes, and its critical position in the Web document were all used together to determine the degree of significance of each extracted word.

1.7 Thesis Outline

The remainder part of this thesis consists of four chapters. They organized as follows:

1. Chapter Two is dedicated to illustrate the relevant subjects surrounding the proposed text based information retrieval system
Chapter One

General Introduction

(TBIR). Descriptions for all theoretical concepts concern the implementation of the proposed work are given.

2. **Chapter Three** introduces the layout of the proposed text based information retrieval system. Descriptions of the algorithms developed to implement the system are also given. It also illustrates the search technique adopted to search for "word list vectors" in the database. Finally, it shows how the retrieval process is performed.

3. **Chapter Four** presents the results of the conducted tests to investigate the system performance behavior.

4. **Chapter Five** gives a list of conclusions derived from analysis of the test results. Also, in this chapter some suggestions are introduced to illustrate the relevant research directions that the work may take in the near future.
Chapter Two
Mining Based on Text Content

2.1 Introduction

The purpose of Text Mining is to process unstructured (textual) information, to extract meaningful numeric indices from the text, and, thus, to make the information contained in the text accessible to the various data mining algorithms [Dav08]. All words found in the input documents will be indexed and counted in order to compute a table of documents and words. This table is represented as a matrix of frequencies that enumerates the number of times that each word occurs in the processed document. This basic process can be further refined to exclude certain common words (i.e. stop words) and to stem the words. However, once a table of extracted unique words from documents is established many of standard statistical and data mining techniques can be applied to derive dimensions or clusters of words or documents [Mic04].

In this chapter, some of the concepts and methodologies that are commonly used in text mining process are introduced briefly. The understanding of these relevant issues will pave the way to significance of the main stages of the introduced IR system.

2.2 Knowledge and Information

Data are discrete facts that are represented in some symbolic way and are recorded in many different formats. By themselves, data may have little meaning or use. While Information means data being used and interpreted by a person. The data are relevant and have a purpose in influencing or altering a person’s decisions or behavior. Knowledge is a personal or organizational experience, valued and insight, gained from an applied use of information. It is actionable information; its availability in the right place, time, and context could help the person to make right decision. Together,
these terms (i.e., Data, Information and Knowledge DIKs) are fundamental to every organization’s daily existence. They used closely together to build the foundation for knowledge management (KM) and for the use of Web mining. Also, the machine learning tools used for data mining that use this triple to extract the exact knowledge [Ada05].

2.3 Information Retrieval

Information retrieval (IR) is the science of searching for documents, for information within documents, metadata about documents, as well as for searching the existing relational databases in the World Wide Web. There is overlap in the usage of the terms "data retrieval", "document retrieval", "information retrieval", and "text retrieval". Each has its own body of literature, theory, praxis, and technologies. IR is interdisciplinary science; it is based on computer science, mathematics, library science, information science, information architecture, cognitive psychology, linguistics, statistics, and physics.

Automated information retrieval systems are used to reduce what has been called "information overload". Many universities and public libraries use IR systems to provide access to books, journals and other documents. Web search engines are the most visible IR applications [Haz10].

2.3.1 Machine Learning for Information Retrieval

Learning techniques have been utilized in Information Retrieval (IR) applications long before the recent advances of the Web. Some of the research efforts in this area have introduced the use of machine learning in:

- Information extraction,
- Relevance feedback,
- Information filtering,
- Text classification and Text clustering.
A. Information Extraction

Information extraction (IE) refers to the techniques designed to automatically identify useful information from text documents. It has the goal of transforming a collection of documents into information that is more readily digested and analyzed. Table (2.1) illustrates the difference between information retrieval and the information extraction fields [Kra97].

<table>
<thead>
<tr>
<th>Information Retrieval (IR)</th>
<th>Information Extraction (IE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aims to select relevant documents (i.e., it finds documents).</td>
<td>Aims to extract relevant facts from the document (i.e., it extracts information).</td>
</tr>
<tr>
<td>Views text as a list of unordered words.</td>
<td>Interested in structure and representation of the document.</td>
</tr>
</tbody>
</table>

Thus IE works at a finer granularity level than IR does on documents and makes information retrieval more precise. Applications of IE include the summarization of documents in well defined subject areas and automatic generation of databases from text [Kra97].

B. Relevance Feedback

It helps users to conduct searches iteratively and reformulate the search queries using the results of previously retrieved documents. In relevance feedback, a model can learn the common characteristics of a set of relevant documents in order to estimate the probability of relevance for the remaining documents [Fuh91]. Various Machine Learning algorithms, (such as genetic algorithms, ID3, Neural Networks and simulated annealing) have been used in relevance feedback applications [Kra97, Che98].

C. Information Filtering

Information filtering techniques try to learn about users interests according on their evaluations and actions, and then to use this information to analyze new documents. Many personalization and collaborative systems
have been implemented as software agents to help users in their interactions with information systems [Mae 94].

2.3.2 Web Mining

It is the application of machine learning (data mining) techniques to web-based data for the purpose of learning or extracting knowledge. Web mining encompasses wide variety techniques, including soft computing. Web mining methodologies can generally be classified into one of three distinct categories: (i) web usage mining, (ii) web structure mining, and (iii) web content mining. Applying data mining techniques to web page content is referred to as web content mining which is a new sub-area of web mining, built upon the field of information retrieval [Chu03].

Figure (2.1) illustrates the Web mining architecture in general. Any Web mining system should cover the following subtasks [Ada05]:

1. **Resource finding:** it is the task of retrieving intended Web documents. It focuses on a one-time analysis of web sites and cannot deal with constantly changing web sites (like, news sites and stock exchange sites) where the information is constantly added or modified.

2. **Information selection and pre-processing:** automatically select and pre-process specific information from the retrieved Web resources. It consists of two sub tasks: (i) selection of interesting data from the downloaded web documents, and (ii) transformation of the selected data into a formal representation (in order to construct formal representation tables).

3. **Generalization:** automatically discovers general patterns at individual Web sites as well as across multiple sites. Many of data mining techniques use "association rules, clusters and classification trees with rules to accomplish this kind of tasks”.

4. **Analysis:** Validations and/or interpretation of the mined patterns. The user may be supported by graphical interfaces.
2.3.3 Retrieval vs. Mining

Retrieval tasks focus on retrieving relevant existing data or documents from a large database or document repository. While mining tasks focus on inferring new information or knowledge in the data. Data retrieval techniques are mainly concerned with improving the speed of retrieving data from a database, whereas data mining techniques analyze the data and try to identify interesting patterns (knowledge). However, it should be noted, that the distinction between information retrieval and text mining is not clear. Many applications (such as; text classification and text clustering) are often considered both as information retrieval and as text mining [Ant05].

2.3.4 Web Data Types

Different data types exist in the Web environment, they are [Ada05]:

1. **Content data:** The presented data in the Web pages (i.e. the data conveyed by Web page to be presented to the users). This usually consists of text and graphics, but not limited to only these two forms of data.

2. **Structure data:** It is the data used to describe the organization of the content. It includes Intra-page structure information and inter-page structure information; the first covers the arrangement of various HTML
or XML tags within a given page, this can be represented as a tree structure, for example the <html> tag. While the principal kind of inter-page structure information is hyper-links which connecting one page to another.

3. **Usage data:** It is the data that describes the pattern of usage of Web pages, such as IP addresses, page references, and the date and time of accesses.

4. **User profile data:** It is the data that used to provide demographic information about users of the Web site; it includes registration data and customer profile information.

### 2.3.5 Web Mining Applications

Web mining includes three kinds of applications, they are: (i) structure mining, (ii) content mining, and (iii) usage mining.

#### A. Web Structure Mining

The goal of Web structure mining is to generate structural summary about the Web site and Web page. Technically, Web structure mining tries to discover the link structure of the hyperlinks at the inter-document level. It is based on the topology of the Hyperlinks, Web structure mining categorizes the Web pages and generates the information, such as the similarity and relationship between different Web sites.

Web structure mining can have another aim that is "discovering the structure of Web document itself". This type of structure mining can be used to reveal the structure (i.e. schema) of Web pages; this makes it possible to compare/integrate Web page schemes. This type of structure mining will facilitate the use of database techniques for accessing information in Web [Ada05].
B. Web Usage Mining

Web servers, proxies, and client applications can quite easily capture data about Web usage. Web server logs files contain information about every visit to the pages hosted on a server. Some of the useful information includes what files have been requested from the server, when they were requested, the Internet Protocol (IP) address of the request, the error code, the number of bytes sent to the user, and the type of browser used. Web servers can also capture referrer logs, which show the URL of the page from which a visitor had made the next request [Ada05].

Client-side applications, such as Web browsers or personal agents, can also be designed to monitor and record user’s actions by performing analysis on Web usage data (sometimes it is referred to as clickstream analysis).

Web mining systems can discover useful knowledge about a system’s usage characteristics and the users’ interests. This knowledge has various applications, such as personalization and collaboration in Web-based systems, marketing, Web site design, Web site evaluation, and decision support. Figure (2.2) illustrates the Web usage mining.

![Web usage mining diagram](His10)
C. Web Content Mining

Web content mining implies an automatic search for information resources available online. In web content mining the actual contents of web pages is examined, most often the text contained in the pages, and then performs some knowledge discovery procedure to learn about the pages themselves and their relationships. Most typically, this is done to organize a group of documents into related categories. It is especially beneficial for web search engines, since it allows users to more quickly find the information they are looking for in comparison to the usual "endless" ranked list [Ant05].

The Web document usually contains several types of data, such as text, image, audio, video, metadata and hyperlinks. Some of them are semi-structured (such as, HTML documents) while others have more structured data (such as the data in tables or database attached to dynamically generated HTML pages). But, most of the existing data content has unstructured nature. The unstructured characteristic of Web data forces the Web content mining tasks to move towards more complicated approaches [Sho09]. Web content mining systems mainly imply some tasks that belong to information retrieval and text mining (e.g., information extraction, text classification and clustering, and information visualization). Also, it includes some new applications, such as knowledge discovery on the Web [Bae99].

2.4 Traditional Information Retrieval System

An information retrieval system is an application that stores and manages information about document contents, often textual documents but possibly multimedia. The system assists users in finding the information they need. It does not explicitly return information or answer questions. Instead, it informs on the existence and location of documents that might
contain the desired information. Information retrieval system can support three basic processes: the representation of the content of the documents, the representation of the user's information need, and the comparison of the two representations [Djo09]. Figure (2.3) presents the main steps of a typical classical IR system.

The comparison of query against the document representations is called the matching process. The matching process usually results in a ranked list of documents. Users can walk down the document list looking for the information they need. Ranked retrieval hopefully puts the relevant documents towards the top of the ranked list, minimizing the time the user has to invest in reading the documents. The simplest but effective, ranking algorithms use the frequency distribution of terms over documents, and they may use some statistics over other information (such as the number of hyperlinks that point to the document). Ranking algorithms based on statistical approaches easily halve the time the user has to spend on reading documents [Enr09].

![Diagram of Classical Information Retrieval System](image)

Fig. (2.3) Classical information retrieval system [Djo09]
2.4.1 Performance Measures

The measures usually employed to evaluate the performance of an information retrieval system are precision and recall. Precision ($Pr$) is “How many of the retrieved results are relevant” and the recall ($Re$) is “How many of the relevant results are retrieved”; mathematically, they defined as [Wil95]:

\[
Pr = \frac{R}{N}, \qquad \text{(2.1)}
\]

\[
Re = \frac{R}{M}, \qquad \text{(2.2)}
\]

Where,

$R$ is the total number of the relevant retrieved items.
$N$ is the retrieved items.
$M$ is the relevant items.

Equivalently, the measures $Pr$ and $Re$ can be defined as:

\[
Pr = \frac{t_p}{t_p + f_p}, \qquad \text{(2.3)}
\]

\[
Re = \frac{t_p}{t_p + f_n}, \qquad \text{(2.4)}
\]

<table>
<thead>
<tr>
<th>Relevant</th>
<th>Non relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieved</td>
<td>true positives ($t_p$)</td>
</tr>
<tr>
<td>Not retrieved</td>
<td>false negatives ($f_n$)</td>
</tr>
</tbody>
</table>
Equations (2.3) and (2.4) with table (2.2) clarifies the relationships between the precision and the recall with the contingency parameters.

From the contingency table (2.2) it is noticed that there are two actual classes, (i) relevant and (ii) non relevant. An information retrieval system can be thought of as a two-class classifier which attempts to label them such that it retrieves the subset of documents which it believes to be relevant [Enr09].

### 2.4.2 Major Information Retrieval Models

For the information retrieval method to be efficient, the documents are typically transformed into a suitable representation. There are several representation models; these models describe the IR process in a formal way [Wil95]. There are three ‘classic’ retrieval models: (i) the Boolean model, (ii) the vector space model and (iii) the probabilistic model. In all these models, a list of keywords describes the content of a document. For this purpose, it is fundamental that the index terms describe the content of the document specifically. Thus, words which appear in nearly every document have no importance for the IR process (stop words), while some infrequent words may be considered significant. Hence, a numeric weight $W_{ij}$ is assigned to each keyword(or index term). Definition-1 constitutes the base for the IR models described next [Bae99].

---

**Definition(2.1) Weight of an index term [Bae99]**

"Let $t$ be the number of index terms in a system and $k_i$ be a generic index term. $K = \{k_1, ..., k_t\}$ is the set of all index terms. A weight $W_{ij} > 0$ is associated with each index term(keyword) $k_i$ of a document $d_j$. For an index term which does not appear in the document text, $W_{ij} = 0$. With the document $d_j$ is associated an index term vector represented by $D_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$. Further, let $g_i$ be a function that returns the weight associated with the index term $k_i$ in any $t$-dimensional vector (i.e., $g_i(D_j) = w_{ij}$)."
Chapter Two  

Minning Based on Text Content

A. Standard Boolean Model

The Boolean model is one of the first introduced IR models. Queries are represented by Boolean expressions, this means that a query is connected by NOT, AND, OR. The model differs whether an index term is absent or not. Therefore, the weight for an index term is \( w_{i,j} \in \{0, 1\} \) [Boo85, Cat87, Coo88, Pai84, and Sal83]. The following definition sets the properties of a Boolean model.

**Definition (2.2) The Boolean Model [Bae99]**

"For the Boolean model, the index term weight variables are all binary i.e., \( w_{i,j} \in \{0, 1\} \). A query \( q \) is a Boolean expression. Let \( q_{dnf} \) be the disjunctive normal form for the query \( q \). Further, let \( q_{cc} \) be any of the conjunctive components of \( q_{dnf} \). The similarity of a document \( d_j \) to the query \( q \) is defined as:

\[
\text{sim}(d_j, q) = \begin{cases} 
1, & \exists q_{cc} \left( q_{cc} \in q_{dnf} \right) \land \left( \forall k, g_i(d_j) = g_i(q_{cc}) \right) \\
0, & \text{otherwise}
\end{cases}
\]

if \( \text{Sim}(d_j , q) = 1 \) then the Boolean model predicts that the document \( d_j \) is relevant to the query \( q \) (it might not be). Otherwise, the prediction is that the document is not relevant."

The definition shows the main drawback of the Boolean model. It is not possible to denote a partial match which can lead to retrieve too few or too many documents. That implicates that the model is not able to list documents according to their ranks. Apart from that, its simple mathematical formalism is its main advantages [Bae99].

B. Extended Boolean Model

The goal of the extended Boolean model is to overcome the drawbacks of the Boolean model that has been noticed in information retrieval.
applications. The Boolean model doesn't consider weights in queries, and the result set of a Boolean query is often either too small or too big. The idea of the extended model is to make use of partial matching and weights as in the vector space model. It combines the characteristics of the Vector Space Model with the properties of Boolean algebra and ranks the similarity between queries and documents. According to this model a document may be somewhat relevant if it matches some of the queried terms and will be returned as a result, whereas in the Standard Boolean model it wasn't. Thus, the extended Boolean model can be considered as a generalization of both the Boolean and vector space models [Sal83]. One of the extended Boolean models uses Fuzzy Set theory in which an element has a varying degree of membership to a set instead of the traditional binary membership choice. The weight of an index term for a given document reflects the degree to which this term describes the content of a document. Hence, this weight reflects the degree of membership of the document in the fuzzy set associated with the term in question. The degree of membership for union and intersection of two fuzzy sets is equal to the maximum and minimum, respectively, of the degrees of membership of the elements of the two sets. The Boolean operators are softened by considering the query-document similarity to be a linear combination of the min and max weights of the documents [Edw86].

C. Vector Space Model (VSM)

The vector space model is the most common model in IR. Opposite to the Boolean model, real numbers are assigned as weights $W_{ij}$ to the index terms in documents and user queries. For the calculation of the similarity, these weights are used. Documents can be ranked by the obtained similarity values [Mic04]. The following definition clarifies the main idea of VSM:
**Definition (2.3). The Vector Space Model [Bae99]**

"For the vector model, the weight $w_{i,j}$ associated with a pair $(k_i, d_j)$ is positive and non-binary. Further, the index terms in the query are also weighted. Let $w_{i,q}$ be the weight associated with the pair $[k_i, q]$, where $w_{i,q} = 0$. Then, the query vector is defined as $\bar{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$. Where, $t$ is the total number of index terms in the system. As before, the vector for a document $d_j$ is represented by $\bar{d} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$.

The similarity between a document and a query is a real number (i.e. $\text{sim}(d_j, q) \in [0, 1]$). To rank the documents, a user enters a query. Then the similarity of the query with all documents is measured and finally, the documents are presented to the user in a ranked order.

Relevance rankings of documents in a keyword search can be calculated by comparing the deviation of angles between each document vector and the original query vector; where the query is represented using same kind of representation vector as the documents. The commonly used VSM metric is depicted in Figure (2.4). It is easier to calculate the cosine of the deviation angle between the two vectors, instead of the angle itself as stated in the following equation [Djo09].

$$C(Q, D) = \frac{\sum_{j=1}^{m} q_j d_j}{\sqrt{\sum_{j=1}^{m} q_j^2 \sum_{j=1}^{m} d_j^2}} \quad \text{................................. (2.6)}$$

Where,

$\bar{q} = \{q_1, q_2, \ldots, q_m\}$, the representation vector of the query.

$\bar{d} = \{d_1, d_2, \ldots, d_m\}$, the representation of the document.
Chapter Two

The vector space model has the following advantages over the standard Boolean model [Bae99]:

1. It is a simple model is based on linear algebra.
2. The "term weights" in VSM are not binary.
3. It allows the computing of continuous degree of similarity between queries and documents.
4. It allows ranking for database documents according to their possible relevance.
5. It allows partial matching.

D. Probabilistic Model

The last classic model uses probabilistic theory to compute the relevance of documents depending on the user query. In order to calculate the required parameters of the relevant document set, after an initial guessing, a feedback from the user is required which differentiates between relevant and non-relevant documents [Bae99].

2.4 Stemming

Stemming of words is an important pre-processing text operation step; it should be handled before the indexing step of input documents. The term "stemming" refers to the reduction of words to their roots so that; different grammatical forms or declinations of verbs are identified and
indexed (counted) as the same word. The goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance: car, cars, car’s, cars’ will be car [Bau03].

2.4.1 Morphology

Morphology is the study of how root words and affixes are composed to form words. Morphology can be divided into inflection and derivation:

- Inflection is the form variation of a word under certain grammatical conditions. In European languages, these conditions consist notably of the number, gender, conjugation, or tense.
- Derivation combines affixes to an existing root or stem to form a new word.

Derivation is more irregular and complex than inflection. It often results in a change in the part of speech for the derived word [Pie06].

2.4.2 Krovetz Stemming Algorithm

It is a 'light' stemmer, as it makes use of inflectional linguistic morphology. Although English is a relatively weak morphological language, languages such as Hungarian and Hebrew have stronger morphology where thousands of variants may exist for a given word. In such a case the retrieval performance of an IR system would severely be impacted by a failure to deal with such variations. The Krovetz stemmer effectively and accurately removes inflectional suffixes in three steps: (i) the conversion of a plural to its single form (e.g. ‘-ies’, ‘-es’, ‘-s’), (ii) the conversion of past to present tense (e.g. ‘-ed’), and (iii) the removal of ‘-ing’. The conversion process firstly removes the suffix, and then passes though a process of checking in a dictionary for any recoding, and finally returns the stem to a word [Kro93].
2.5 Fuzzy Logic

Fuzzy concepts are derived from fuzzy phenomena that commonly occur in the real world. For example, rain is a common natural phenomenon that is difficult to describe precisely since it can rain with varying intensity, anywhere from a light shower to a torrential downpour. Since the word rain does not adequately or precisely describe the wide variations in the amount and intensity of any rain event, so, "rain" is considered as fuzzy phenomenon [Meh03].

2.5.1 Fuzzy Set Representation

The encapsulation of objects into a collection whose members all share some general features is naturally implies the notion of a set. Sets are used often and almost unconsciously; we can talk about a set of even numbers, positive temperatures, personal computers, fruits. For example, a classical set A of real numbers greater than 6 is a set with a crisp boundary, and it can be expressed as: \( A = \{ x \mid x > 6 \} \).

Unlike the aforementioned example of conventional sets, a fuzzy set expresses the degree to which an element belongs to a set. The characteristic function of a fuzzy set is allowed to have values between 0 and 1, which denotes the degree of membership of an element in a given set. If X is a collection of objects denoted generically by X, then a fuzzy set A in X is defined as a set of ordered pairs:

\[
A = \{ x, \mu_A(x) \mid x \in X \}, \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots
\]

Where, \( \mu_A(x) \) is called the membership function for the fuzzy set A. The membership function maps each element of X to a membership grade (or membership value) lays between 0 and 1. Usually X is referred to as the universe of discourse it may consist of discrete objects or continuous space. This can be clarified by the following example: Let \( X = \{ 0, 1, 2, 3, 4, 5, 6 \} \) be a set of the number of children a family may choose to have, then the
fuzzy set \( A = \) "sensible number of children in a family" as illustrated in Figure (2.5) may be described as follows [Ana06]:

\[
A = \{(0,0.1),(1,0.3),(2,0.7),(3,1),(4,0.7),(5,0.3),(6,0.1)\}
\]

Or, equivalently:

\[
A = \{(0.1/0),(0.3/1),(0.7/2),(1/3),(0.7/4),(0.3/5),(0.1/6)\}
\]

Another representation of the characteristic function is:

\[
A = \sum_{x_i \in X} \frac{\mu_A(x_i)}{x_i} \quad \text{..........................................................(2.8)}
\]

![Figure (2.5) An example of the membership grades of the fuzzy set [Ana06]](image)

The representation of the above equation is for the discrete universe of discourse. While, the fuzzy set representation for the continuous membership function is given as follows:

\[
A = \int \frac{\mu_A(x)}{x}, \quad \text{..........................................................(2.9)}
\]

2.5.2 Membership Function (MF)

The function that tries a number of each element \( x \) from the universe is called the membership function[Jan93] In real-world applications of fuzzy sets the shape of membership functions is usually restricted to a certain class of functions which can be specified with only a few parameters.
Chapter Two

Mining Based on Text Content

The most well known (MFs) are triangular, trapezoidal, and Gaussian; Figure (2.8) presents the commonly used shapes for membership functions [Meh03].

A. Trapezoidal Membership Functions

The trapezoidal membership function (T) is specified by four parameters \{a, b, c, d\} as stated in the following equation [Meh03].

Figure (2.6) illustrates the shape of the trapezoidal MF and the arrangement of its four parameters.

\[
T(x : a, b, c, d) = \begin{cases} 
0, & x \leq a \\
\frac{(x - a)}{(b - a)}, & a \leq x \leq b \\
1, & b \leq x \leq c \\
\frac{(d - x)}{(d - c)}, & b \leq x \leq c \\
0, & d \leq a 
\end{cases}, \quad \ldots \ldots (2.10)
\]

Fig.(2.6) Trapezoidal Membership Function[Ana06]

B. Triangular Membership Functions

The triangular membership function is specified by three parameters \{\alpha, \beta, \gamma\} as stated in the following equation [Meh03].
\[ \Lambda(x; \alpha, \gamma, \beta) = \begin{cases} 
0, & x \leq \alpha \\
\frac{(x - \alpha)}{(\beta - \alpha)}, & \alpha \leq x \leq \beta \\
\frac{(\alpha - x)}{(\beta - \alpha)}, & \alpha \leq x \leq \beta \\
0, & x > \gamma 
\end{cases}, \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (2.11) \]

Figure (2.7) illustrates the arrangement of the triangular MF’s parameters, where \( \alpha < x < \beta \), and they determine the x coordinates of the three corners of the underlying triangular-membership function [Meh03]. Figure (2.9) illustrates this function.

![Triangular membership function](image)

C. S-shaped Membership Functions

The s-shaped membership function is specified by three parameters \( \{\alpha, \beta, \gamma\} \) as stated in the following function [Meh03].

\[ S(u; \alpha, \gamma, \beta) = \begin{cases} 
0, & x \leq \alpha \\
\frac{2(x - \alpha)}{(\beta - \alpha)^2}, & \alpha \leq x \leq \beta \\
\frac{1 - 2(\alpha - x)}{(\beta - \alpha)^2}, & \alpha \leq x \leq \beta \\
1, & x > \gamma 
\end{cases}, \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (2.12) \]
Figure (2.8) illustrates the arrangement of the S-shaped MF’s parameters.

![Fig.(2.8) S-shaped Membership Function [Ana06]](image)

D. Gaussian Membership Functions

The Gaussian-membership function is specified by two parameters \( \{c, \sigma\} \), as stated in the following function [Zad88]. Figure (2.9) illustrates this function.

\[
G(x : c, \sigma) = \exp \left( - \frac{(x - c)^2}{2\sigma^2} \right) \quad \text{.........................(2.13)}
\]

Where,  
C represents the membership-function center (or the mean, value) and \( \sigma \) is the standard deviation value.

![Fig.(2.9) Gaussian Membership Function [Zad88]](image)
2.5.3 Fuzzy Set Operations

Union, intersections, and complement are the most basic operations in classic sets. Corresponding to the ordinary set operations, fuzzy sets too have similar operations, which were initially defined by Zadeh (the founder of the fuzzy-set theory) [Aga05]. Some of these operations are:

1. The union of two fuzzy sets (A and B) is a fuzzy set C, written as:
   
   \[ C = A \cup B \text{ or } C = A \lor B; \]
   whose membership function \( \mu_c(x) \) is related to those of A and B. As pointed out by Zadeh, a more intuitive but equivalent definition of the union of two fuzzy sets A and B is the "smallest" fuzzy set containing both A and B. As shown in the following equation:

   \[
   \mu_c(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \lor \mu_B(x), \forall x \in X, \text{.......... (2.15)}
   \]

2. The intersection of fuzzy sets can be defined analogously. The intersection of two fuzzy sets (A and B) is a fuzzy set C, written as:
   
   \[ C = A \cap B \text{ or } C = A \land B, \]
   whose membership function is related to those of A and B, as in the case of the union of sets, it is obvious that the intersection of A and B is the "largest" fuzzy set that is contained in both A and B. As is shown in the following equation:

   \[
   \mu_c(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \land \mu_B(x), \forall x \in X, \text{..........(2.16)}
   \]

3. Complement of a fuzzy set A, denoted by A', is defined by the following membership function:

   \[
   \mu_{A'}(x) = 1 - \mu_A(x), \text{.......................................................... (2.17)}
   \]

   Figure (2.10) illustrates the union, intersection and complement operations of fuzzy logic.
2.6 Hashing

One of the main challenges facing the data indexing tasks is to define and implement a mapping from a domain of keys to a domain of locations. The domain of keys can be any data type (strings, integers, and so on). The mapping between these two domains should be both quick to compute and compact to represent. The main goal is to avoid collisions, where collision occurs when two or more keys map to the same location. If no keys collide, then locating the information associated with a key will simply be the process of determining the key's location. Whenever a collision occurs, some extra computations are necessary to further determine a unique location for a key. Collisions therefore degrade performance [Cha86].

Assuming that domain of keys has \( N \) possible values and the domain of location has \( m \) values; then collisions are always possible whenever \( N > m \), that is, when the number of values exceeds the number of locations in which they can be stored. The best performance is therefore achieved by having \( N = m \), and using a 1:1 mapping between keys and locations. Defining such a mapping is easy. It requires only a little knowledge of the representation of the key domain. Figure (2.11) illustrates a simple example of the hashing process; it is clear that all the elements belong to the same class has been hashed with the others and the results are indexed to represent all the set's values [Meh82].
The choice of a hash function is extremely important. The ideal function (or perfect) hash would distribute all elements across the buckets such that no collisions ever occurred. The most important consideration in choosing a hash function is the type of data to be hashed. Data sets are often biased in some way, and a failure to account for this bias can ruin the effectiveness of the hash table. For example, the hash table shown in Figure (2.12) is storing integers that are all powers of two. If all keys are to be powers of two, then \( H(k) = k \mod 5 \) is a poor choice: no key will ever map to bucket 0 [Cha86].

To create a hashing function for a hash table it is often the used function should have larger domain. To create an index from the output of the function a modulo can be taken to reduce the size of the domain to match the size of the array; however, it is often faster on many processors to restrict the size of the hash table to powers of two by applying bits mask operation.

![Fig. (2.11) An example of hash function [Meh82]](image)

![Fig. (2.12) Visual illustration of a hash table [Meh82]](image)
2.7 Bitwise Operations and Masking

Bitwise operators operate on an integer variable at an individual bit level. They may be used on signed, unsigned and char variables, but for indexing purpose it is preferred to be used on unsigned integer types. A number is represented in a computer by a series of bits - giving the number in binary (base 2). Each digit tells whether the relevant power of 2 is present or not [Joh05]. In computers, bits are usually grouped together. Table (2.3) illustrates some standard sizes.

<table>
<thead>
<tr>
<th>Name</th>
<th># of Bits</th>
<th># of Bytes</th>
<th>Range in Dec.</th>
<th>Range in Hex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>nibble</td>
<td>4</td>
<td>0.5</td>
<td>0-15</td>
<td>0-0XF</td>
</tr>
<tr>
<td>byte</td>
<td>8</td>
<td>1</td>
<td>0-255</td>
<td>0-0XFF</td>
</tr>
<tr>
<td>word</td>
<td>16</td>
<td>2</td>
<td>0-65,535</td>
<td>0-0XFFF</td>
</tr>
<tr>
<td>double word</td>
<td>32</td>
<td>4</td>
<td>0-4,294,967,295</td>
<td>0-0XFFFFFFFF</td>
</tr>
</tbody>
</table>

When working with bytes it is possible to perform the bitwise operations corresponding to each of the basic Logical operations AND, OR & XOR. The logical operation is applied on each pair of bits taken from the two inputs and from the same position.

Bitwise operations are particularly useful for masking. In this case, each bit in a byte represents a value which may be either ON or OFF (i.e. true or false). This is particularly common when accessing hardware devices at a low level. For example a bit is turned on by a bitwise OR (|) with 1 in the relevant position. Also, the test concerned with whether a bit is set could be done by a bitwise AND (&) with 1 in the relevant position. A bit turned off by a bitwise AND (&) with NOT (~) 1 in the relevant position [Wil03].

2.8 Web Media Elements

The World-Wide Web (WWW) hinges on three enabling protocols: (i) the Hyper Text Markup Language (HTML) that specifies a simple markup language for describing hypertext pages, (ii) the Hypertext Transfer Protocol
Mining Based on Text Content

Chapter Two

(HTTP) which is used by web browsers to communicate with web clients, (iii) and Uniform Resource Locators (URL's) which are used to specify the locations of documents in WWW [Sum01].

Many types of Servers are used in the Internet world e.g. Mail Server, Domain Name System (DNS) Server, Dynamic Host Configuration Protocol (DHCP) Server, Web Server, Database Server, etc. Every server has its own applications and services. But the most popular Server in Internet is the Web Server [Pol05]. By far the most popular Web Server program is Apache. It is claimed that Apache servers are used to host more than 68% of all web sites on the Internet [Chr09].

Web Server is a special type of Servers used to post the web pages requested by the Web Browser. Its duty is not only posting the requested pages but also, compiles and executes the Server side code if it is embedded in the requested web page. Server side code contains logical decisions, evaluations, performs calculations, accesses database and manipulate data. Usually Server side code is written using C# and/or Java languages (in Dot Net and Java technologies, respectively).

Web Browser is a client software (or application) used to request the web pages. Browsers not only used to request the webpage; they also work as interpreters. They interpret the HTML and display the Webpage contents. Beside to interpreting HTML codes, they can, interpret the client side code if it is embedded in the web page [Lee04].

The communication protocol at application layer between Web Browser and Web Server is HTTP protocol. To establish the connection between Web Server and Web Browser, HTTP uses port 80. HTTP uses two methods (i.e. GET and POST) to request web Pages. To understand all the above terminologies, it is necessary to understand the usual communication between Web Server and Web Browser.

35
The communication steps of HTTP protocol are as follows [Chr09]:

1. A Web Browser requests a web page (using GET or POST method of HTTP).
2. The Web Server processes the request and responds by sending the requested web page.
3. The Web Browser interpreters the Webpage and displays it.
4. After getting the Webpage, the user enters some inputs and clicks the submit button.
5. But before submitting the Webpage, Web Browser interprets the client side code; if some requirements are not fulfilled by the user; it issues a prompt message asking the user to complete those requirements.
6. If all validations are complete, then the Web browser resends (called POSTBACK) the Webpage to Web Server.
7. The Server processes the code embedded in the Webpage and processes all the instructions given in the Server-side code, then sends the Webpage to the Web Browser again.
8. The Web Browser again interpreter the web page and displays it.

Figure (2.13) illustrates steps taken by HTTP to handle a Webpage.

The client-server model is the simplest distributed software architecture; it divides the processing activities between clients and servers. The distributed software applications have one of the following distributed architectures:

A. **Two Tier Architecture**

It is the simplest distributed applications architecture, where a direct communication takes place between client and server. There is no intermediate getaway between client and server. The two tier architecture is composed of (i) Data tier (Database) and (ii) Client application (client tier) as shown in figure (2.14) [Pol05].
Chapter Two

Mining Based on Text Content

Fig. (2.13) The communication steps between Web server and Web browser [Chr09]

Fig. (2.14) Two tier architecture [Pol05]
B. Three Tier Architecture

3-Tier client-server architecture has 3 essential components (i) Client, (ii) Middleware, (iii) and Database server). The additional tier (i.e. middle tier server) is between the client user interface (client) and the server (data management) components. This middle tier provides process management, where business logic and rules are executed, and it can accommodate hundreds of users by providing functions such as queuing, application execution and database staging [Ber04]. Figure (2.15) illustrates the architecture of typical 3-tier model.

![Three tier architecture diagram](image)

Fig. (2.17) Three tier architecture [Pol05]

The benefits of using 3-tier architecture are [Chr09]:

1. It is easier to modify or replace any tier without affecting the other tiers.
2. Separating the application and database functionality means better load balancing.
3. Adequate security policies can be enforced in each tier.

C. N-Tier Architecture

In the 3-tier applications, the middle tier is generally not a monolithic entity (e.g. a program), it is implemented as a collection of components initiated by several client-initiated business transactions. One component can call other components for help in order to implement a request; some components may also act as gateways which may encapsulate legacy applications running on mainframe [Ber04].

2.9 Document

A document is one of the units of information. A document can be an article, a book, or images extended with text. In terms of its physical representation, a document can be represented as any physical unit like a file, an email or a Web site. Furthermore, a document may consist of different kinds of media; these media can be images, videos, sounds, or text.

A document has syntax and a structure settled by a person or a document generator. The syntax can be expressed implicitly in its content but also through a declarative language or, referring to code, by programming languages. Additionally, a document has semantics specified by the author. The semantics may differ according to its use. The presentation style of a document specifies how the document content is presented to users, for example on the screen or on printing paper [Chi01].

Markup languages, like HTML or XML allow sophisticated representations of documents. Therefore, Web documents need not be considered as atomic entities, but as interrelated objects which can be indexed and retrieved separately. So, the structure of a document can be used to perform IR in all components of a document. Figure (2.16) illustrates the typical representation of a Web document [Bae99].
Fig(2.16) Characteristics of typical Web document [Bae99]

2.9.1 Hyper Text Mark-up Language (HTML)

HTML is an evolving language, and each new version is given a number. The first definitive version was originally developed by Tim Berners-Lee at Conseil Européen pour la Recherche Nucléaire (CERN), and was popularized by the Mosaic browser developed at National Computer Security Association (NCSA). During the course of the 1990s it has blossomed with the explosive growth of the Web. During this time, HTML has been extended in a number of ways. The Web depends on Web page authors and vendors sharing the same conventions for HTML. This has motivated joint work on specifications for HTML.

HTML 2.0 was developed under the aegis of Internet Engineering Task Force (IETF) to codify common practice. HTML+ at 1993 and HTML 3.0 has been proposed much richer versions of HTML. Despite never receiving consensus in standards discussions, these drafts led to the adoption of a range of new features. The efforts of the WWW Consortium (W3C)'s HTML Working Group to codify common practice in 1996 resulted in HTML 3.2. In 1998 HTML 4 was issued and then some enhancements are done on this version in 1999 to produce the HTML 4.1 which is currently used and is supported by all of the Web browsers.

Each version has attempted to reflect greater consensus among industry players, so that the investment made by content providers will not be wasted
and their documents will not become unreadable in a short period of time. HTML has been developed with the vision that all manner of devices should be able to use the existing information on the Web (e.g., PCs with graphics displays of varying resolution and color depths, cellular telephones, hand held devices, devices for speech for output and input, computers with high or low bandwidth, and so on)[Jak07].

HTML4.1 language has a number of characteristics which are:
1. Publish online documents with headings, text, tables, lists, photos, etc.
2. Retrieve online information via hypertext links, at the click of a button.
3. Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
4. Include spread-sheets, video clips, sound clips, and other applications directly in their documents [Jak07].

2.9.2 HTML Documents Structures

HTML files have the extension of .htm, .html, or .shtml. HTML documents have two parts, the head and the body. The body is the larger part of the document, as the body of a letter you would write to a friend would be. The head of the document contains the document's title and some other relevant information, and the body contains most everything else. HTML document's consisting of tags, enclosed in angle brackets (like <html>). HTML tags normally come in pairs like <h1> and </h1>. The first tag in a pair is the start tag, the second tag is the end tag they are also called opening tags and closing tags; in between these tags web designers can add text, tables, images, etc [Spi04].

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting
structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts in languages such as JavaScript which affect the behavior of HTML WebPages [Jak07]. In the file illustrated in this Figure there is a header and a body. The template can be used to build another HTML files. This code contains the tags that are important to format the content of a document in a particular format and also have imported an external CSS file; which will be described later.

```html
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <meta name="GENERATOR" content="Arachnophilia 3.9">
  <meta name="description" content="Comprehensive Documentation and information about HTML.">
  <meta name="keywords" content="HTML, tags, commands">
  <title>The CTDP HTML Guide</title>
  <link href="style.css" rel="stylesheet" type="text/css">
  <!-- Background white, links blue (unvisited), navy (visited), red (active) -->
</head>

<body>
  <center><h1>HTML Document Structure</h1></center>
  <p>This is a sample HTML file.</p>
</body>
</html>
```

*Fig.(2.17) An illustrative example of an HTML file*

HTML has three key attributes [Chm02]:

1. **Link-ability**: Data is hyperlinked in HTML, letting one piece carry you to another.
2. **Simplicity**: HTML is simple, making it easy to learn and to display.
3. **Portability**: HTML is stripped down, making it portable over networks, operating systems and languages.
However, HTML is limited in [Chm02]:

1. **Intelligibility**: How well data knows itself.
2. **Adaptability**: How well data changes in response to environmental changes.
3. **Maintainability**: How easily data is maintained.

XML overcomes limitations of HTML and other markup languages, while providing capabilities that are not a part of the earlier languages. XML has two very important features: (i) it allows Webpage designers to create their own tags according to their needs, (ii) XML completely separates content from formatting through the use of style sheets, that means there is no data in an XML document contained in tags that only give formatting instructions. Rather, all the data is identified by the XML tags in which it is enclosed. Pages that will be written in XML will be much easier for the computers to read and to catalog, making search engines more powerful and the Web become more useful tool.

Though XML does have many benefits, there are still a few drawbacks:

(i) Only Internet Explorer 5.0 can read XML files directly, the older versions of browsers would have to convert it into HTML before viewing the document.

(ii) XML is not very user friendly when you make mistakes like HTML is (i.e., it is sensitive about upper and lower case letters, quotation marks, and closing tags) [W3c08].

### 2.9.4 JavaScript

JavaScript is a scripting language mainly used for writing dynamic Web pages. When a script written in JavaScript is embedded in a Web page, it will be executed by the Web browser on the client machine. JavaScript and the Java programming language both use syntaxes influenced by that of C syntax. JavaScript copies many Java names and naming conventions; but the two languages are unrelated and have very different semantics [Rob10] as programming languages.
JavaScript has the following unique features [Lou10]:

1. JavaScript is a scripting language: that is the source code is interpreted, instead of compiled and executed.
2. JavaScript is a dynamic typing language: Data types are associated with values, instead of variables. In other words, variables are declared without specific data types. They can be assigned with values of any data type.
3. JavaScript is a prototype based language: Prototypes, instead of classes, are used for defining object properties, methods, and inheritance.
4. JavaScript uses associative arrays to represent objects: Property names and values are stored as associative array elements. Properties and their values can be added, changed, or deleted at run-time.
5. JavaScript supports functions as first-class functions - Functions are objects. Like regular objects, functions can be created during execution, stored in data structure, and passed to other functions as arguments.

An example of a JavaScript code that inserts another JavaScript code into the HTML document is shown in Figure (2.18) [Rob10].

```html
<html>
<!-- HelloNestedScripts.html
Copyright (c) 2008 by Dr. Herong Yang, http://www.herongyang.com/--> 
<head><title>Hello from Nested Scripts</title></head>
<body>
<pre>
<script type="text/javascript">
document.writeln("Hello World!");
document.writeln("<script type="text/javascript">");
document.writeln("document.writeln(" Hello World! ");");
document.writeln("</script>");</script>
</pre>
</body>
</html>
```

**Fig. (2.18) JavaScript code**
2.9.4 Cascaded Style Sheet (CSS)

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation semantics (i.e. the look and formatting) of a document written in a markup language. It is most common language used to style web pages written in HTML and eXtended HTML XHTML, but the language can also be applied to any kind of XML document, including plain XML, Scalable Vector Graphics (SVG) and XML-based User-interface Language (XUL) [Mcb10]. CSS is designed primarily to enable the separation of document content written in HTML or a similar markup language from document presentation, including elements (such as the layout, colors, and fonts). This separation can:

(i) improve content accessibility,
(ii) provide more flexibility and control in the specification of presentation characteristics,
(iii) enable multiple pages to share formatting, and
(iv) reduce complexity and repetition in the structural content.

While the author of a document typically links that document to a CSS, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified [Gla11].

CSS style information can be either attached as a separate document or embedded in the HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium [Nit10].

The CSS syntax is declared by rules, it has two main parts: a selector and one or more declarations, as shown in figure (2.18):
The selector is normally the HTML element wanted to be styled, and each declaration consists of a property and a value. The property is the style attribute wanted to be changed, and each property has a value [mcb10], Figure (2.19) presents an example illustrates the CSS File structure [Gla11]:

```
<head>
<title>Title Goes Here </title>
<style type="text/css">
#left-col p {
  color: #222;
  font-weight:bold;
}
</style>
</head>
```

CSS has the following three styles:

1. External: is a separate page which is then linked to the web page. Therefore, the styles are External to, or Outside of, the Web Page.

2. Internal (or embedded) Style: is placed in the <head> section of the current page, and is applied to an element on that current page only. These styles cannot be reused on other web pages. The <style> tag is used to write an Internal Style. Figure (2.23) presents an example of "internal style" code.

3. Inline: is placed directly inside an HTML element in the code. The Style Builder can not be used to make an Inline Style. Instead, to create an inline style we should go into the HTML code and type the style their. The inline CSS has no Selector. Because any inline style embedded directly inside the HTML element it styles, so, there is no need for a selector. An Inline style defeat the purpose of using CSS and negates most, if not all, of CSS's advantages, like the separation of content from presentation.
2.9.5 Unicode Transformation Format-8 (UTF-8)

UTF-8 is a character encoding scheme that can be as compact as ASCII if the file is just plain English text but can also contain any Unicode characters (with some increase in file size). So, UTF-8 possesses the advantages of being backward-compatible with ASCII. UTF-8 has become the dominant character encoding for the World-Wide Web, accounting for more than half of all Web pages. The Internet Engineering Task Force (IETF) requires all Internet protocols to identify the encoding used for character data, and the supported character encodings must include UTF-8 [Joh03].

The Internet Mail Consortium (IMC) recommended that all e-mail programs must be able to display and create mail using UTF-8. UTF-8 is also increasingly being used as the default character encoding in operating systems, programming languages, APIs, and software applications. UTF-8 encodes each of the specified code points in the Unicode character set using one to four 8-bit bytes (termed “octets” in the Unicode Standard). Code points with lower numerical values (i.e., earlier code positions in the Unicode character set, which tend to occur more frequently in practice) are encoded using fewer bytes, making the encoding scheme reasonably efficient. In particular, the first 128 characters of the Unicode character set, which correspond one-to-one with ASCII, are encoded using a single octet with the same binary value as the corresponding ASCII character [Pik03].

Most Western European languages require less than two bytes per character. For example; characters from Latin-based scripts require only 1.1 bytes on average. Greek, Arabic, Hebrew, and Russian require an average of 1.7 bytes. Finally, Japanese, Korean, and Chinese typically require three bytes per character [Yer03]. The ASCII characters are represented by themselves as single bytes that do not appear anywhere else, which makes UTF-8 work with the majority of existing APIs that take bytes. This removes the need to write a new Unicode version for every API [Pik03].
Chapter Three
Design and Implementation of TBIR system

3.1 Introduction

This chapter is dedicated to present the layout of the proposed and established Web document retrieval system. The introduced system is named "Text Based Information Retrieval System (TBIR)". In the first part of this chapter the system layout is illustrated. Next, the system enrollment operations are presented, including the text operations. A focus on the developed stemming algorithm is given; this algorithm was enhanced to better suite the TBIR main goal. Also, the implemented steps of other operations (like the cleansing, removing stop words, tokenization, hashing, and sorting) are clarified.

In the next part of this chapter, the steps taken in the stage of matching between the feature vectors extracted from a query document and those stored in a database (which belongs to one of the existing HTML documents in the server) are given. Fuzzy logic based criteria are used to decide the matching degree between each pair of HTML documents. The ranking of the database documents is accomplished according to the fuzzified matching results. Finally, the method used to speed up the matching between two lists (or records) of key words is clarified.

3.2 The Architecture of TBIR System

This research aims to build a content-based information retrieval system that supports a client-side Web query to retrieve all similar Web documents according to their topic relevancy with the query document. At the enrollment stage, the extracted lists (or records) of keywords from the Web pages are stored in the system's database. Each document is associated
with its own list of keywords which is considered as its descriptive information.

During the retrieval stage (i.e., after the enrollment phase) the queried Web is analyzed to extract its list of keywords. Then, the extracted list of keywords is matched with the pre-extracted and archived lists in the database; to decide the degree of relevancy between the queried Web document and the Web documents (which are represented in the database). Figure (3.1) illustrates the architecture of the system.

![Fig. (3.1) The proposed TBIR system architecture](image)

### 3.3 Client and Server Architecture

TBIR is a sort of recognition process; it is composed of two phases:

(i) The enrollment (off-line) phase: it includes the operations conducted on server side only.

(ii) The retrieval (on-line) phase: some of its operations are conducted on client side, other operations accomplished at server side's
Figure (3.2) illustrates the operation conducted on both sides (i.e., server and the client side).

3.4 Enrollment Phase

This phase is server perspective, because there is no client interaction during this phase. So, it is an offline process. The extracted list of keywords from each HTML document is stored in the database with the URL address of that Web HTML document. Figure (3.3) illustrates the operations involved in the enrollment phase. It involves the following tasks:
A. Preprocessing

As any data mining application, preprocessing/data cleansing operation is an obligatory step. Stemming is apart of the preprocessing tasks, taking into consideration that the quality of the whole retrieval process depends performance of stemming step. From the conducted experiments it is noticed that preprocessing is the most time-consuming stage. It accommodates a convenient environment for the next stages to be applied on a suitable data forms without anomalies or redundancies. This stage doesn’t mean only noise elimination as will be seen later, where in this context noise means the data that are not textual or the text that is not actually viewed. The advantages of preprocessing stage are: (i) lead to fast retrieval results due to reduction of document representation to a small number of keywords, (ii) improves the search result by elimination of commonly used words.
The preprocessing stage covers the following operations:

**A. Downloading and Conversion:** It implies the conversion of UTF-8 code of the Web pages to the corresponding traditional computer representation of characters (i.e., the ASCII code). The downloading and code conversion is accomplished by defining an object from the class System.Net.WebClient(). This class provides common methods for sending data to and receiving data from a resource identified by an URL which belongs to Net name space. The object derived from this class has methods can deal mainly with Web pages. In our developed system the method used to download the HTML source code is the DownloadData () method. The parameter fed to this method is the URL value of the wanted web page. This method downloads the text Web contents of specified URL as a System.Byte () array, the returned array of bytes contains the Web text content encoded using UTF-8 encoding. The function GetString(), belongs to UTF8Encoding class, is used to decode all the UTF-8 string to ASCII string.

**B. Tokenization and Filtering:** After downloading the Web documents and transforming the text content from UTF-8 code to ASCII code, they have to be decoded further but this time from the document's tagged source code to a pure plain text; this means removal of the mark-up tags, JavaScript and CSS codes. The conversion to pure plain-text format will make the primitive text operations operating on this plain.

After the code conversion step the plain should be tokenized, the tokenization operation converts a sequence of characters stored in a string block to a finite set of understandable words stored in an array of strings. Tokenization leads to throwing away certain set of text symbols, including punctuations and digits from the plain.

In addition to the above steps, another additional step should be accomplished during the tokenization operation; it is the trimming (i.e. removal of the leading white spaces), because the leading characters
prevents the positive boolean matching between two equal words, even if they are explicitly equal.

Algorithm (3.1) illustrates the implementation of the above mentioned steps. The implemented steps in algorithm (3.1) covers the following tasks:

1. **Transform the text to lower-case characters:** as a form of character unification; this step ensures that all the matches instances are correct.

2. **Removal of alphanumeric and special character:** all special characters are removed, but some of them are considered as text delimiters. For example, the words separated by hyphen ("-") are considered compound-words and are not broken.

3. **Strip the tags:** removes the HTML tags from the source code.

4. **Removal of structural codes:** the structural codes are used to design the document and format its contents in a particular style. They should be removed because they are not contextual data. They have nothing to do with the actual text in the documents; so, they are erased completely because the established system is concerned with Web content mining application. The removed structural codes are the following:

   1. **CSS language codes:** which may come in the HTML source code in three forms (i.e., inline, external, and internal forms):
      
      a) The inline is imbedded inside the HTML tags themselves. It is already removed when the tag is stripped.
      
      b) The internal CSS code is enclosed inside the <style></style> tags, so these tags are stripped with their inside code using a regular expression.
      
      c) The external CSS file is referenced in the HTML source code using the <link> instead of <style>. Anyhow this tag is removed in a way similar to inline CSS code; since it is already exist within the header tag, when this tag stripped the CSS code is erased as well.
2. JavaScript: It is like CSS codes. JavaScript is added to the HTML source code in order to add additional interaction between the website and its visitors. So, it concerns the design and the view of the document. The JavaScript codes are enclosed within the `<script>..............</script>` tags.

Algorithm (3.1) tokenization and filtering

**Goal:** Convert the HTML source code to its plain. And truncating this plain to separated words.

**Input:** The HTML plain.

**Output:** An array of words corresponds to the input text.

**Step 1:** Remove the java script and the css file sheet from the HTML source code by applying the regular expression "<style.*?>(.|
)*?</style>" "<script.*?>(.|
)*?</script>". This means that every thing between these tags is removed.

**Step 2:** Remove the HTML tags but store the occurrence of some tags in the text to be used later. The tags are removed using this regular expression "<[^>]*>".

**Step 3:** Convert the text to a lower-case.

**Step 4:** Remove the numeral characters and special characters using this regular expression: "[^\sa-zA-Z\-_]", which means to not take any character except the alphabetical characters and hyphen.

**Step 5:** this step involves the following tasks:

- Prepare an empty array
- Set two pointers(i) front=1; which is considered the beginning of the word and the(ii) rare =1; which is the end of the word, get the HTML plain text

- While rare <> 0 and front < length of plain text do
    - Set rare = front + position of the delimiter
    - If rare <> 0 Then
        - Set tie= rare- front
        - Set token = series of characters from (tie - front)
        - Set array (index) = token
        - Set front = rare + 1
    - End If
- Index=index+1
- End While

- Array (index) = token
B. Text operations

The established TBIR works on a textual content data of the HTML documents, this imposes that text operations are inevitable, (namely, these operations belong to the text mining field). The TBIR has a text analyzer that analyses the extracted by implementing certain textual mining operations. The implemented text operations are:

A. **Stop-words removal**: The excluded stop words are put in a predefined exclusion dictionary (or table) to exclude such terms from the extracted HTML plain text because this kind of words occurred too frequently and their existence is insignificance for mining tasks. So, they are not worthy to be indexed in the system's database. Typical stop words include "a", "and", "an", "the", "but", and so on. From the experiments it is seen that the removal of the stop words helps in increasing the accuracy of the results and, also, it helps in reducing the size of the database. Algorithm (3.2) illustrates the removal of stop words.

```
Algorithm (3.2) Stop words removal

Goal: to remove commonly used words from the input HTML plain text.
Input: input text representing the extracted HTML plain text.
Output: a text that is stop-words-free

Step 1: Call algorithm (3.1) // to filter the text and tokenize the words.
Step 2: Load the stop_word.TXT file to the "unwanted_words()" array.
Step 3: For Each word in unwanted_words() do
   Check if word exists in the plain text then remove this word wherever it exists in the text.
Next
Step 4: return the stop-word-free text to be passed to another operations. (Go to stemming)
```

B. **Stemming**: There are nine parts of speech, (i.e., nouns, verbs, adjectives, adverbs, pronouns, conjunctions, prepositions, interjections and articles). As a result of the stop words removal five parts of speech has been
totally removed, they are: the pronouns, conjunctions, prepositions, interjections and articles. The other parts of speech are also checked for some of stop words to be also removed before stemming. If the remaining words didn't match with the stop words; they are stemmed out and then rechecked against the stop words. The main concern of the proposed stemming is to keep up the words relatively meaningful after stemming to the old one, and do not transform the words from one part of speech to another since this may contribute to change the word's meaning.

In general, stemming deals with derivational forms and inflectional forms; the applied stemming algorithm deals with inflectional forms only unlike other algorithms (e.g. porter's algorithm) which deal with both. The inflectional morphology doesn’t affect the stem while the derivational affects the stem and may change the meaning as described in the previous chapter. Anyhow, during the conducted tests, it is noticed that stemming the derivational forms is not convenient to be used in the proposed TBIR since it ruins a large number of informative keywords to forms which are semantically correct but become opaque words and autonomous (i.e. have no relation with other words exist in the same article). For example, the words end with "ize", "ise", "en", ify" for verbs and "tion", ary", "ssion" for nouns must remain as they are; but the derivational stemming makes such words further fine-granulated, such that they become unsuitable as keywords, and consequently they negatively affect the quality of the information that expected to be retrieved by TBIR.

On the other hand, the inflectional stemming involves the removal of all suffixes added to words as a result of grammatical importance, like the progressive, past tenses for verbs and plural for nouns. Their removal does not change the stemmed word's part-of-speech, this means that the name remains a name, the verb remains a verb. Furthermore inflectional stemming has nothing to do with the affixes; for example the "un" prefix that's added to the word to negate it remains as it is. Table (3.4)
illustrates an example of the inflectional vs. derivational stemming. Algorithm (3.4) illustrates the implemented stemming rules in details.

The stemming alone is not yet enough in processing the words to originate them to their roots; there is another operation which is considered a complementary to the stemming, it is called the lemmatization and used for checking the stem. If the produced stem is not grammatically correct, the lemmatization will add or remove what the stem needs to be an integrated word. If the implemented lemmatization is not correct, or not considered at all, many of the stemming results will become false negatives/positives which can effectively cause a defect in the performance of retrieval system because a lot of tokens will be deformed. Not every word needs to be lemmatized; for instance, the suffixes of "reading", "reads" will change to get the stemmed form "read" standing for the infinitive "to read"; in this case both the lemmatized word form and the word stem are equal.

<table>
<thead>
<tr>
<th>Table (3.1) Examples of stemming types</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(a) Inflectional stemming</strong></td>
</tr>
<tr>
<td>inflectional</td>
</tr>
<tr>
<td>computers</td>
</tr>
<tr>
<td>organizations</td>
</tr>
<tr>
<td>purifies</td>
</tr>
<tr>
<td>whitened</td>
</tr>
<tr>
<td>serializing</td>
</tr>
</tbody>
</table>

The proposed stemming algorithm is extendable; it can be extended to include other rules. In the reconstructed TBIR the rules mentioned in algorithm (3.4) had led to satisfactory level of word granularity, further stemming rules may defragment the words, and would degrade the information retrieval performance. It is noteworthy that the applied algorithm is somehow similar to Korvtz algorithm; because it deals with the inflectional forms only. But it uses a positive dictionary to assure words correctness. Besides stemming steps, algorithm (3.3) implies the implemented lemmatization steps.
Algorithm (3.3) Stemming

Goal: To originate the words to their original form, to remove the suffix and compensate the word with the original suffix and integrates the word to its generic form. The resulted word must be semantically correct and not changed from its original part-of-speech to another.

Input: Array of vowel letters \{a, e, u, o, i\}, array of consonant letters, letter set \{j, k, c, q, w, x, y, v\}, and the HTML plain text.

Output: A string of words inflectionally correct and relatively meaningful to the proposed TBIR.

Step 1: For each word apply the following special rules:

- If word ends with (ing) and its (length ≤ 4) Then return the word and don’t stem.
- Else If word ends with (ed) and its (length = 3) Then return the word and don’t stem.
- If the stem ends with ll and the word ends with (ing) and its length with stem ≤ 8 Then just remove (ed, ing).
- If stem ends with ll and (length ≤ 7) and ends with "ed" Then just remove (ed, ing).
- If the stem ends with (dg) Then remove the suffix (ed, ing) and add ("e")
- If word ends with (ural) Then remove the (al) and add (e)
- If word ends with (ies) or word ends with("ied") Then replace the syllable with (y)
- If word ends with (es) and stem ends with \{ ch, x, sh, s, ss\} Then remove (es).
- If stem ends with \{x, w, eed, or, er, y, ok, ap, ain\} Then just remove the suffix.
- If word contains one of these diphthongs (ea, ee, ou, ai, oi, io, ei, ie ) Then just remove the suffix.
- If word ends with (ely, ness) Then just remove the suffix.
- If word ends with (ified or ifies) Then substitute (ied, ies) with (y)

Step 2: Impose the following generic rules:

Ing rule:
- If (word length > 5) and (word ends with \{ing, ed\}) Then just remove the suffix
- If word (lastletter - 1) ≠ word(lastletter - 2) and word (lastletter - 1) ∈ consonant And word(lastletter - 2) ∈ vowel Then word = word ∪ "e".
- If word (lastletter - 1) = word(lastletter - 2) Then remove word (lastletter - 1) from the word //to remove final char if repeated

S rule:
- If (word length ≠ 3) and word ends with(s) and not ((word ends with (us, ss, os, cs, is)) or not(word ∈ vowel () and word not capitalized Then don’t remove (s) and return the word as it is.

Step 3: Concatenate all the words that satisfied the previous stemming and lemmatization rules and those that didn’t satisfy them, all together in a string variable called resulted_string.
C. Word Frequency Determination

After tokenization, removal of the special characters and alphanumeric characters, stemming and removal the stop words, then the stage of determining the frequency of occurrence of each extracted word is applied. The list of words' frequency of occurrence will be used later for weighting the degree of significance of each of the words. Algorithm (3.4) illustrates the implemented steps of this process.

Algorithm (3.4) word frequency

Goal: To find the number of times each word appeared in the text and to associate this frequency number with its word.

Input: Preprocessed text.

Output: The frequency number of each word appeared in the text.

Step 1: Open an empty array of records; contains two elements: the word and its frequency.

Step 2: For Each word In the list () do

- If word length > 12 or word length <= 2 Then continue the loop and take another word.
- Else count the number of occurrence of the tested word (i.e., determine how many times the word is found in the array). The count result is stored as an integer, called X for each word. the frequencies of all words produce a vector of integer values (i.e. \( X = \{w1,x1],[w2,x2]........wn,xn]\) . Where n is the number of keywords and w is the corresponding word.

- Each word has been put in the word () array will not be traversed again and directly go to step 2 to take the next listed word.

Next {word in the list}

Step 3: Add the contents of words () vector into a Data table representing each word and its frequency of occurrence.
D. Words Attributes Determination

This stage involves the detection of important keywords which can be utilized to define the identity of the processed HTML file. It is already known that the mark-up languages embed tags within its source code and those tags have no added contextual values, they only affect the text appearance when a Web browser is used to view the document content. For example, the tags block <bold>RAM</bold> leads to the appearance of the word "RAM" as bold style. Usually, this kind of text style tags is used as indicators to highlight the significance of each word connected to these tags. So, from these structure-base indicators the results of the retrieval system may become sharper than taking the term frequency alone to determine the word's weight.

Usually, any article discusses a particular subject must present the keywords of that subject in different form with respect to other words, this will help the reader to quickly allocate the important keywords of the article and will make him able to further search about the article subject according to those keywords. So, they are considered as search indicators (guides) help the reader to find similar articles. Anyhow, when the reader tries to use those keywords in his search manually, his trails to get similar documents are not always successful, and if they succeeded the retrieved documents may be similar but irrelevant. Irrelevancy may happen because the electronic content is always growing and the search space gets larger. This means that the recall and precision within such huge volumes of data are not always a goal. So, some measures based on intelligency and rationality are needed as well. This means that the retrieved documents have to be informative, relevant and rationally acceptable by the user. Using Automated retrieval systems search for the similar documents without user intervention (i.e. without need for user assessment of what are the important keywords). In such automated systems the allocation of keywords that may lead to better retrieval results is left to the system;
which should "mine" huge textual content of a document in order to extract the required useful knowledge that may lead to more relevant documents.

Table (3.2) shows the list of selected words features. The first and the second columns are for the extracted words and their determined frequency of occurrence, which are originated by algorithm (3.4). The other columns are to register the corresponding print style features. Algorithm (3.5) is used to fill each column with boolean values “true” or “false” in front of each word. The features used in the proposed TBIR are "Bold/Strong", Italic, Hyperlink, Title, Capitalization, Underline and Header"; only this set of features is considered because they most commonly used to highlight the words significance and relevancy. The implemented TBIR system is designed to accept further word's print style features. Algorithm (3.6) illustrates the calculation of the feature's weight (significance factor). There is another usage for these features; at the retrieval stage they used for scoring this will be discussed in details later in this chapter.

<table>
<thead>
<tr>
<th>Word</th>
<th>Tf</th>
<th>Title</th>
<th>Bold</th>
<th>Hyperlink</th>
<th>Underline</th>
<th>Italic</th>
<th>Header</th>
<th>Total</th>
<th>Weight</th>
<th>Final Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td>30</td>
<td>true</td>
<td>a₁</td>
<td></td>
<td></td>
<td>true</td>
<td>a₂</td>
<td></td>
<td>F=Σa₁</td>
<td>H=ΣTF/N</td>
</tr>
<tr>
<td>Word 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E. Determination of Words' Significance Factor

After allocating the significant words, the result is a list of records each holds the word's name, frequency of occurrence and a set of Boolean values representing the print style attributes of the extracted word. Each attribute extracted is given a certain weight value according to its common usage for highlighting the keywords. For example: title=2.1, header =1.1 and bold=1.2. When some word satisfies this triple then its attributes weight
is set 4.3. As a result this weight is combined with its frequency weight factor to produce the overall weight value.

Algorithm (3.5) words’ feature detection

Goal: To allocate significance degree (i.e., weight) of the extracted and stemmed words from the HTML source code, remarking every word when it appears as [bold, italic, title, header, hyperlink, underlined].

Input: The HTML source code.

Output: A set of Boolean vectors, each vector represents a specific feature, and the length of these vectors is equal to the list size of the extracted keywords. Each boolean value in this vector represents the existence or absence of the feature in a specified keyword.

Step 1: Store each pattern matches the expression: "<tag>[^>](.|
)*?</tag>" from the input HTML text.

Step 2: Strip the tags from this pattern, using the expression "<[^>]*>", to substitute each found instance of these characters with null. The remaining is a word (or words) styled with a feature called "styled word".

Step 4: Set the counter = 0

Step 5: For counter = 0 to length of keyword list do.

If styled word = list (counter) Then add “true” to the data table.

Table (counter) (column) // table (3.5)

Else add “false”

Next

Algorithm (3.5) is applied six times; each time it checks one of the considered print style features; except the Capitalization which has its own algorithm since it is not indicated by HTML tags.

In English text it is very likely to find words that are fully capitalized this indicates that this word is either an abbreviation or an acronym, Algorithm (3.7) detects words that are totally uppercased to also allow this propriety to contribute in weighting the word that satisfies this property.
**Algorithm (3.6) Significance factor finding.**

**Goal:** To search the values of each vector extracted from algorithm (3.6), when a "true" value is found it is substituted by a constant otherwise add nothing, these constants accumulated with the corresponding once (on the same row) to find the print style weight of each word.

**Input:** The extracted boolean feature vectors resulted from algorithm (3.6).

**Output:** A numeric feature vector corresponds to the input boolean vectors.

**Step 1:** Set \( N = \) number of the input boolean vector

**Step 2:** For \( i = 0 \) to keywords list do

\[
\text{accumulator} = 0
\]

For \( j = 0 \) to \( N \) do

If data table \( (i) (j) = \) (true) Then

\[
\text{accumulator} += \text{constants (j)}
\]

result_array \( (i) = \) (accumulator)

Data table \( (i) (10) = \) accumulateur

Next

Next

**Algorithm (3.7) Capitalized word detection.**

**Goal:** To detect every word that is totally capitalized.

**Input:** original HTML plain text before transforming it to a lower case and making the tokenization.

**Output:** a feature vector representing the capitalization.

**Step 1:** Call algorithm (3.2) // to remove the stop words

**Step 2:** Call algorithm (3.3) // to stem the input text.

**Step 1:** Set list_counter = 0

**Step 2:** For Each (word) In keyword list /// it is in lowercase already.

transform the (word) to upper case

If there is any match between this word and the input HTML plain text then set Data table (list_counter) (column) = true // where column=8

\[
\text{list_counter} += 1
\]

// this data table used to hold all of the feature vectors extracted in front of their corresponding keywords.

Next
It is clear from the algorithm that this feature it is not detected as is done with the other features extracted using algorithm (3.6), because it doesn't depend on the mark-up tags, so it has to be coded separately.

F. Determination of the Overall Weight

After the HTML plain text is parsed, using the text analyzer, and its keywords are extracted, the weight of each word is computed. This weight consists of two parts: (i) frequency weight, and (ii) textual feature weight. To ensure the Web text size invariance, the normalized word frequency (i.e., probability of occurrence) is determined. The last three columns in table (3.5) are concerned with the weighting information, they are:

1. **Word ratio**: is the relational number of repetitions of each keyword in the extracted set of words; it is determined using the following mathematical expression:

   \[
   F_n(j) = \frac{F(j)}{\sum_{i=1}^{n} F(i)} \times 100
given by (3.1)
   \]

   Where,

   \(F(j)\) is the frequency of occurrence of the Jth keyword,

   \(F_n(j)\) is the normalized frequency, i.e. probability, of occurrence.

   \(N\) is the total number of extracted keywords.

2. **Textual feature weight** (\(w_f\)): it is a real value equal to the sum of the individual weights of each existing textual (i.e., printing style) feature which are allocated using algorithms (3.5) (3.6), (3.7). It is calculated using the following expression:

   \[
   w_f(j) = \sum_{i=1}^{g} w_i(j)
given by (3.2)
   \]
Chapter Three

Design and Implementation of TBIR

Where,

\[ w_i(j) \] is the corresponding weight of \( i^{th} \) feature of the extracted \( j^{th} \) word. The value of the weight \( w_i(j) \) is determined using the mapping function:

\[
W_i(j) = \begin{cases} 
\alpha, & \text{If the } i^{th} \text{ feature of } j^{th} \text{ keyword is} \\
0, & \text{Otherwise}
\end{cases} \quad \ldots\ldots\ldots\ldots\ldots(3.3)
\]

3. **Overall-weight:** this value is gotten from the multiplication of the pervious two weights (i.e., \( F_n \) & \( W_f \)), that is:

\[
W_{Overall}(j) = F_n(j) \times W_f(j) \quad \ldots\ldots\ldots\ldots\ldots(3.4)
\]

Where, \( W_{Overall}(j) \) is the overall weight value of the \( j^{th} \) keyword.

**G. Keyword Sorting**

The extracted keyword list has to be sorted in descending order according to the corresponding overall weight of each keyword in the list. Then the premiere subset of keywords which has the largest overall weights is selected and the rest is ignored. Bubble sort is used to sort the weights in a descending order. Algorithm (3.8) illustrates the implemented steps of the sorting process. This algorithm is used to sort the keywords according to their overall weights to ensure that only the highly weighted (i.e., significant) words are used as keywords and stored in the retrieval database.

The experiments show that taking few keywords is not fruitful, it is better to increase the number of the keywords, up to a certain extent, to increase the system's recognition capability. So, in this work the number of selected keywords is set to 150 maximum, since the weights after 150 elements will be small, and it can take less than this number as will be seen in chapter 4 where this number is considered one of the system parameters.
H. Hashing (Feature Vector Coding)

Accessing information based on a key is a vital operation in information retrieval. Hashing is a very common technique for storing/accessing data in such a way that the data can be retrieved very quickly. In the proposed TBIR the hashing technique is applied after the preprocessing, weighting and sorting stages. It is used to encode the Boolean values of each keywords' textual features and assemble them as a single integer value. This helps in reducing the storage area, providing a fast access. At the retrieval stage the hashed feature vector is decoded to check the associated textual features of each keyword.

The used hash function depends on the logical masking principle explained in chapter 2; the index number of each textual feature in the list is represented as a bit. The resulted bits are accumulated, to produce a binary chain of bits, and as a result this chain is represented by a byte value; the assembling process could be represented using the following equation:

\[
\text{Algorithm (3.8) bubble sort algorithm.}
\]

**Goal:** To sort the words records according to their overall weight values.

**Input:** The overall weight () vector.

**Output:** A list of ordered words which are sorted in a descending way.

<table>
<thead>
<tr>
<th>Step 1: For ( i = 0 ) to number of keywords -2 do</th>
</tr>
</thead>
<tbody>
<tr>
<td>For ( j = i + 1 ) to keywords ()-1 do</td>
</tr>
<tr>
<td>If ( \text{weight} (i) &lt; \text{weight} (j) )</td>
</tr>
<tr>
<td>Then</td>
</tr>
<tr>
<td>swap(( \text{weight} (i) ), ( \text{weight} (j) ))</td>
</tr>
<tr>
<td>Swap (keywords (i), keywords (j))</td>
</tr>
<tr>
<td>Next</td>
</tr>
<tr>
<td>Next</td>
</tr>
</tbody>
</table>

| Step 2: If the length of the list () is >\( \rho \) then discard the list () // \( \rho \): is a constant value.

---

Algo 3.8: Bubble Sort Algorithm

**Goal:** To sort the words records according to their overall weight values.

**Input:** The overall weight () vector.

**Output:** A list of ordered words which are sorted in a descending way.

**Algorithm:**

1. **Step 1:** For \( i = 0 \) to number of keywords -2 do
   - For \( j = i + 1 \) to keywords ()-1 do
     - If \( \text{weight} (i) < \text{weight} (j) \) Then
       - swap(\( \text{weight} (i) \), \( \text{weight} (j) \))
       - Swap (keywords (i), keywords (j))
   - Next
   - Next

2. **Step 2:** If the length of the list () is >\( \rho \) then discard the list () // \( \rho \): is a constant value.
Chapter Three

Design and Implementation of TBIR

\[ \text{Hash}(j) = \sum_{i=1}^{8} 2^i f(B_i(j)), \] \hspace{2cm} (3.5)

Where,

\( B_i(j) \) is the boolean value of \( i^{th} \) textual feature of \( j^{th} \) keyword.

\( f() \) is a mapping function (from boolean to integer) such that:

\[
f(b) = \begin{cases} 
1 & \text{If } b = \text{true} \\
0 & \text{If } b = \text{false} 
\end{cases}
\] \hspace{2cm} (3.6)

Algorithm (3.9) illustrates the encoding process. Figure (3.4) shows an illustrative example of the hashing process applied on a textual feature set. The resulted hash value is decoded at the scoring and retrieval phase.

![Binary encoding for a set of textual features](image)

To retrieve (decode) the actual boolean value of each feature the "And" bitwise masking operation is utilized, as expressed in the following expression:

\[
f(b) = \begin{cases} 
\text{true} & \text{If } (\text{hash}(j) \& 2^i)<>0 \\
\text{false} & \text{Otherwise} 
\end{cases}
\] \hspace{2cm} (3.7)
Figure (3.5) shows clearly that the used bitwise hashing method doesn’t have the problem of collision, since each probable set of features has its own unique hash value.

**Algorithm (3.9) Hashing a set of features**

**Goal:** To obtain a hash vector represents the textual feature sets extracted by algorithm (3.5).

**Input:** A set of Boolean values of textual features of each feature set (representing the 3rd columns to 10th of table (3.5)).

**Output:** a hash table.

**Step1:** \( m=\)number of extracted keywords, \( n=\)number of feature vectors.

**Step2:** Code the input value

For \( i = 0 \) to \( m \) do

1. \( \text{counter} = 0 \)

For \( j = 0 \) to \( n \) do

2. If \( \text{data table}(i)(j) \neq \text{false} \) Then
   
   3. \( \text{counter} = \text{counter} || 2^j \)

Next

4. \( \text{Result}(i) = \text{counter} \)

Next

**Fig. (3.5) Decoding using the "AND" bit-wise operation**

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Decoding (ANDing): testing each bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000000000</td>
</tr>
<tr>
<td>1</td>
<td>0100000000</td>
</tr>
<tr>
<td>2</td>
<td>0010000000</td>
</tr>
<tr>
<td>3</td>
<td>0000000000</td>
</tr>
<tr>
<td>4</td>
<td>0000000000</td>
</tr>
<tr>
<td>5</td>
<td>0000010000</td>
</tr>
<tr>
<td>6</td>
<td>0000000100</td>
</tr>
<tr>
<td>7</td>
<td>0000000001</td>
</tr>
<tr>
<td>Hash</td>
<td>1100101111</td>
</tr>
</tbody>
</table>
I. Database Repository

After the accomplishment of the entire above mentioned enrollment stages over all HTML documents hosted in the Web server; then their extracted keywords, weights and their feature's hash values are stored as records in a database at the server side. Each record should consist of an array of subrecords where each subrecord consists of:

1. An extracted keyword.
2. Its overall weight.
3. The hash value of its textual features.

The number of subrecords belong to each record is set 150; which is equal to the number of the extracted significant keywords from each Web document.

Beside the 150 subrecords each record holds a text field to register the URL address of the archived HTML document file. Table (3.3) illustrates the layout of the database table which holds the extracted keywords and their attributes, where N represents the number of the archived HTML files.

<table>
<thead>
<tr>
<th>Record no.</th>
<th>URL address</th>
<th>1st keyword sub record</th>
<th>2nd keyword sub record</th>
<th>150th keyword sub record</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Name</td>
<td>Weight</td>
<td>Hash</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
After extracting the keywords records and send it to the database. The usage of the record as one entity makes the storing operation more efficient than storing each subrecord separately. Algorithm (3.10) briefs all of the operations involved in the enrollment.

**Algorithm (3.10) the extraction of keywords record from HTML documents**

**Goal:** To extract a record of significant keywords from the input document, the steps of this algorithm are applied on both archived HTML files (during the enrollment phase) and query document (during the retrieval phase).

**Input:** HTML document file.

**Output:** A record of significant keywords.

**Step 1:** Download HTML source code and transform the UTF-8 code to ASCII code.

**Step 2:** Call algorithm (3.1) which cleanse and decodes the HTML source code to convert it to a plain text.

**Step 3:** Call algorithm (3.2) to remove the stop words and call algorithm (3, 3) to stem and lemmatize the plain text.

**Step 4:** Call algorithm (3.4) to determine the frequency of the extracted words (TF), call algorithms (3.5), (3.6) to extract the textual features that bonus up the weight. Then the overall weight of each keyword is determined.

**Step 5:** Call algorithm (3.8) to sort the key words according their overall weight which resulted from the step4 and cut the list to 150 for the enrollment and 40 for the query preparation.

**Step 6:** Call algorithm (3.9) to get the hash integer value of the extracted set of textual features (to enumerate the set of features as single enumerative value).

**Step 7:** Assemble the results of all of the above steps in a single record beside to the URL address of the processed Web page.

**Step 8:** Register the record in the data base. (Note: step 8 is not applied in the query phase).
3.5 Retrieval Phase

This phase involves all previously mentioned operations (except the record storage step), and all these operations are directly conducted at the server side under on-line bases with users. Once the client starts the mining operation over a specific query document, the client side program sends a query request with the URL address of the queried HTML document; then the system, at the server side starts the retrieval operation. This operation involves the following main tasks:

A. Query Document Preparation

This stage holds all the operations mentioned in the enrollment phase with the following two differences:

1. The resulted record from algorithm (3.10) is not stored in the database; rather it is used to be matched with the records already stored at the enrollment.

2. The extracted keywords record consists of 40 subrecords; each belongs to one of the highest weighted 40 keywords found in the query document.

B. Scoring

In this stage, matches between the query keywords record and the keyword records stored in the database are performed. The matching is repeated for all registered records in the retrieval database. The result of each matching instance is named a matching score. The match score is determined as the sum of the overall weights of the common keywords found in both matched records. The matching score is calculated using the following equation:

$$S(m) = \sum_{k_w \in Q \cap DB_m} W_{overall}(k_w | Q) W_{overall}(k_w | DB_m)$$

\[\text{..........................................(3.8)}\]
Where, $k_w$ is a keyword found in both query and $m^{th}$ record of the database. $W_{overall}(k_w|Q)$ is the overall weight of the keyword, $k_w$, found in the query record. $W_{overall}(k_w|DB_m)$ is the overall weight of the keyword, $k_w$, found in the $m^{th}$ record of the database. $S(m)$ is the score of matching between both query record and $m^{th}$ record in the database.

The conducted tests have shown that the above scoring criteria, equation (3.8), show a sort of a document size dependency, also, the existence of few highly weighted common keywords may cause bias in grading results. To handle the problem of size dependency and effectiveness of highly weighted keywords, a non-linear mapping function (i.e., S-shaped function) was used to determine the matching score. In our proposed system the tan-hyperbolic function was used to grade the keywords common existence in both query record and the tested database record. The adopted mapping function is defined as:

$$
\mu(w_k) = 1 + \frac{1}{1 + e^{-2w_k}} \quad \text{.........................}(3.9)
$$

Where, $w_k$ is the sum of the overall weights of the keyword (k) found in both the query and the database record.

So, instead of applying equation (3.8) the total matching score between the two records is determined using:

$$
S(m) = \sum_{k_w \in Q \cup DB_m} \mu(W_{overall}(k_w|Q)W_{overall}(k_w|DB_m)) \quad \text{.........................}(3.10)
$$

Where, $k_w$ is common keyword found in both query and $m^{th}$ record of the database.
$W_{\text{overall}}(k_w | Q)$ is the overall weight of the keyword, $k_w$, found in the query record.

$W_{\text{overall}}(k_w | DB_m)$ is the overall weight of the keyword, $k_w$, found in the $m^{th}$ record of the database.

$S(m)$ is the score of matching between both query record and $m^{th}$ record in the database.

$\mu(W)$ is the non-linear membership function, illustrated in Figure(3.6).

Algorithm (3.11) illustrates the process of finding the score using the above mentioned membership function which maps the values in [0,1] interval.

![S-shaped membership function](image)

**Algorithm (3.11) Scoring**

**Goal:** To determine the matching score using the non linear mapping function.

**Input:** A query record (Q) and a database record (DB), $m$=length of the keyword list of the database document, $n$= length of the keyword list of the query document.

**Output:** The score value of the matching.

**Step 1:**

For $j = 0$ to $m - 1$

For $k = 0$ to $n - 1$

If $Q_{\text{keyword}}(j) = DB_{\text{keyword}}(k)$ Then

$x = Q_{\text{weight}}(j) + DB_{\text{keyword}}(k)$

$Sc = \tanh(x)$ // equation(3.9)

$Sc + = Sc$ // to find the total score

Next $k$

Next $j$

$Rank = c \times Sc + \text{bonus}$

**Step 2:** after the determination of rank values for all listed database records (for some query records) then the rank values are sorted using the steps mentioned in algorithm (3.9).
C. Feature Vector Decoding

After the total matching score is calculated by applying equation (3.10) this value is combined with another parameter, which is called the feature "Bonus" weight. This parameter represents the presence of some styling features of each pair of common keywords. Since the styling features of each keyword is stored as a hash (binary) code value associated with the keyword in both matched records (i.e., query and database records). So, to evaluate the bonus parameter as a first step the hash value is decoded (by applying the AND bitwise operator to revive the Boolean feature vectors again, by masking the bits representing the print styling features).

After reconstructing the Boolean sets again, a match between the styling features, belonging to the two sets, is performed. For example, the word "War" appears in the header, title, bold tags and is capitalized in both matched records; this infers that both records have the same relative interests with respect to this common keyword. From this inference the bonus value becomes higher than other matches who may satisfy two; one or even none of the common styling features. Table (3.4) presents an example to simplify the adopted bonus system.

<table>
<thead>
<tr>
<th>Word: &quot;war&quot;</th>
<th>Query</th>
<th>doc 1</th>
<th>doc2</th>
<th>doc3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>Header</td>
<td>$\alpha$</td>
<td>1</td>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>Bold</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>Capitalized</td>
<td>$\alpha$</td>
<td>1</td>
<td>$\alpha$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>results</td>
<td>$(1+\alpha)^2$</td>
<td>$(1+\alpha)^4$</td>
<td>$(1+\alpha)$</td>
<td></td>
</tr>
</tbody>
</table>

The determination of bonus value could be done by applying the following equation:

$$B(Q,k,f_i) = \prod_{i=0}^{n} b(Q,k,f_i) \prod_{i=0}^{n} b(DB_n,k,f_i), \ldots \ldots \ldots \ldots (3.11)$$

Where,
Chapter Three  Design and Implementation of TBIR

$Q$ is the keywords record extracted from the query HTML document.
$DB_m$ is the $m^{th}$ keyword record loaded from the database.
$k$ is a common keyword found in both $Q$ & $DB_m$ records.
$Q.k.f_i$ is the $i^{th}$ styling feature of $k$ keyword found in the query record($Q$).
$DB_m.k.f_i$ is the $i^{th}$ styling feature of $k$ keyword found in the $m^{th}$ record brought from the database.
$n$ is the number of the selected print styling features.
$f$ is a Boolean value \{true, false\}
$b()$ is a mapping function, given by the following expression:

$$b(f) = \begin{cases} 1 + \alpha & \text{If } f = \text{true} \\ 1 & \text{Otherwise} \end{cases} \quad (3.12)$$

Where, $\alpha$ is a constant; its value is taken less than 1.

Algorithm (3.12) presents the implemented steps to determine the matching bonus weight value added to the score of each common word.

### Algorithm (3.12) Decoding (Matching Bonus)

**Goal:** To decode the hash value (which is determined in algorithm (3.10)).

**Input:** Query record and Database record, $m$= length of the Query record keywords, $n$=length Database record, $f$=number of the selected print style features, pos () is the value represents the bit number for each print style feature with is \{0, 1, 2, 4, 8, 16...128\}

**Output:** bonus values vector of the common words in the two records.

**Step 1:** Set bonus as double = 1
**Step 2:** For $i = 0$ to $f$ do
  For $j = 0$ to $m - 1$ do
    For $k = 0$ to $n - 1$ do
      If $q \_keyword(j) = db \_keyword(k)$ Then
        If $(q \_vector(j) \text{ And pos}(i))$ Then bonus = bonus * $(1 + \alpha)$
        If $(db \_vector(k) \text{ And pos}(i))$ Then bonus = bonus * $(1 + \alpha)$
      End If
    Next
  Next
**Step 3:** Return bonus
D. Ranking

The ranking is considered the last process in the retrieval phase of TBIR. When the list of ranks (due to matching of query record with all database records) is established, it is sorted in descending order with the corresponding URL addresses. Then, the top listed URLs are tabulated in output text file sent as output data frame to the client side. The ranking algorithm is accomplished using bubble sort algorithm; which is illustrated in algorithm (3.8). Figure (3.7) illustrates the whole process of the retrieval phase. After the determination steps of both matching score and matching bonus; the total matching rank is determined using the following equation:

\[ R = C W_{overall} + B \]  

Where,

- \( W_{overall} \) is the determined matching score.
- \( Bonus \) is the bonus value due to the styling features of common keywords.
- \( C \) is a constant used to adjust the significance of \( W_{overall} \).
- \( R \) is a total matching rank.
3.6 Search Technique

The matching of two keyword lists is a many_to_many process; since matching aims to find out the common keywords listed in both records, so it requires mapping between two lists of strings whose elements are not sorted according to alphabetical criteria; instead they are ordered according to their relative significance weights. Taking into consideration that the retrieval operation should be performed as an online process, so the matching task should be done as quickly as possible in order to ensure an acceptable performance of the retrieval process. The user should not wait a long time to get the retrieval answer. So, another important issue must be taken seriously to reduce the time of matching the matching between the query and the database record.

In a simple one_to_one process, it is possible to convert the string to an integer number (hash number) according to the ASCII code of its characters, and then be matched with the hash values of the other side to see whether it is common or not. This matching mechanism could be adopted to handle the matching task between the user names and passwords. But this is not a practical solution to be applied in the proposed TBIR or any other system deal with a large number of records with each contains many subrecords. This would clearly spend a long search time in conversion from one type to another.

On the other hand, exhaustive matching trail is a weak and trivial solution to be followed; because the system will try all of the possible choices until it reaches a positive match. The run time is wasted in the brute force search without taking into account the search efficacy.

The proposed and applied search method in the established TBIR system is named frequency-based method because it depends on the frequency of occurrence of the leading letter of each keyword listed in the database record. The idea of the search is similar to that of searching a word
in a dictionary. The mechanism of the proposed method is to alphabetically sort the keyword list, and the keywords have same leading letter are considered as subsets and they may also be called as search intervals since the search of each input keyword is applied, only, within its corresponding alphabetical interval.

The matching between each query's keyword with the list of database record list is performed according to the following steps:

1. Capture the first letter of the query's keyword.
2. Match this keyword with a convenient subset (the subset of keywords belongs to the examined database record) whose keywords begin with same letter of the query's keyword. In this way the search interval is shrunk to a small subset of keywords.

For example if the search word was "SUMER", then all words begin with s letter are prepared for matching. The search, then is performed over a small set of keywords instead of the whole set. Figure (3.8) illustrates the proposed method in details.

![Fig.(3.8) Frequency-based search and match method](image-url)
The matching process is performed as follows:

1. Calculate the histogram of each letter. For example the histogram of the set \{Cosmos, Comet, Uranus, Asteroids, Star, Spacecraft, Satellite\} is equal to \{(c,2), (u,1), (a,1), (s,3)\}. The histogram determination implies the following steps:
   a) Capture the first letter of each word,
   b) Convert the letter to lower letter case,
   c) Get the ASCII code of this letter,
   d) Subtract it by the ASCII of the "a" to find its cardinal number.

   The cardinal number is considered as a pointer of that letter in the histogram array. Whenever a keyword begins with the same initial letter the pointer goes to same index at the histogram array and adds one to the old content value (this means counting every occurrence of the alphabet letters). The histogram determination task is performed overall keywords listed in the tested database record. The size of histogram array is 26 and of byte data type, each histogram element represents the number of keywords begins with certain the alphabet letter whose cardinal number represents the index of the corresponding histogram element. The resulted histogram array is considered a key table in the next step.

2. After the histogram array is found it is, then, used to calculate a list of pointers, each list element points to the start position of the keywords begin with a certain letter. The histogram elements are used as keys to calculate the corresponding peers in the pointer list since the former elements store the sizes of each subset.

   After the pointers generation step, each pointer is considered as the base address for the corresponding set of keywords which have a certain initial letter. The pointer moves next when a new keyword, which has the corresponding initial letter, appears.
3. Sort keywords: after the determination of the initial pointer list, then it used to sort the list of keywords of the tested database records; such that each keyword in the record is put in a new record at position index equal to the corresponding pointer value, and increment that pointer value by 1.

4. Search: the resulting initial pointer array is used to allocate the search area (i.e., the start and end positions) of each tested keyword belonging to a query record. In such case the matching trail will be bounded to a certain subset rather than searching the whole list of keywords belongs to the tested database record. In other words, the pointer list is used to provide the upper bound and the lower bound of each search interval (subset). Algorithm (3.13) presents the implemented steps to perform the selective matching between a query record and one of the tested database records.

**Algorithm (3.13) frequency based search**

*Goal: To sort the keywords alphabetically.*

*Input: Two lists of keywords $A, B$, $n =$ length of $A$, $m =$ length of $B$.*

*Output: A dynamic list of only common keywords in two lists. The resulted list’s elements used directly in the fuzzy scoring. ("no need to store the results").*

**Step 1:** Calculate the histogram for list $A$

$\text{For } i = 0 \text{ to } n \text{ do}$

$\quad ch =$ first letter in $A(i)$

$\quad \text{histogram}(i) = \text{Asc}(ch) - 97$

$\text{Next}$

**Step 2:** Prepare the array of pointers for list $A$

Set position $(0) = 0$

$\text{For } i = 1 \text{ to } 26$

$\quad \text{Set position } (i) = \text{position } (i - 1) + \text{histogram } (i - 1)$

$\text{Next}$

**Step 3:** The matching between with list $B$ according to the $A$'s Histogram and Position arrays, generated from the previous two steps.
For $i = 0$ to $m$

\[ ch = \text{first letter in each } B(i) \]
\[ Index = \text{Asc}(ch) - 97 \]

For $k = \text{position } (Index) \text{ to position } (Index + 1) - 1$ do

\[ \text{If } A(i) = B(k) \text{ Then call algorithm (3.12)} \]

Next

Next
Chapter Four
Test Results

4.1 Introduction

This chapter is dedicated to present the results of the set of tests conducted on the TBIR, to evaluate its performance and to investigate the effectiveness of the system parameters on performance behavior. A set of HTML document files belongs to 9 classes is used as a test material. Microsoft visual basic .NET.2008 is used to establish a system prototype. The tests are conducted on the environment: Windows XP professional operating system; Siemens Fuji Laptop of Intel Celeron processor 2.00 GHz speed. RAM 1.99GHz, 984 MB Physical Address Extension.

4.2 Test Strategy

During the test phase, the involved system parameters are categorized into two sets: (i) control parameters and (ii) performance parameters. The variation in system parameters can be utilizes to tune the system performance, and the aim of the conducted tests is to find out the proper values. The system control parameters are (i) the size of the database document record \(DB_{\text{recsize}}\), (ii) the size of the query document record \(Q_{\text{recsize}}\), (iii) the weight of matching score relative to the bonus value\(C\). (iv) the increment bonus parameter \(\alpha\) and, (v) the exponent rate of the nonlinear weight mapping function \(R\). As performance parameters the precision and recall coefficients are adopted. Also, the execution time is considered as an important performance parameter so it is calculated in each test run.

The effect of soft computing (i.e., usage of the nonlinear S-function) is addressed in this chapter. Its results are compared with the corresponding results of the hard computing. Finally the text operations
(i.e., stemming and stop words removal) are tested to investigate their effects on the retrieval performance.

The total number of HTML documents is 420 documents which belong to 9 classes. Those documents are preprocessed and their produced records are deposited in the database. Table (4.1) shows the topic classes and the number of used HTML documents belonging to each class. More details about the used documents are given in appendix (A).

<table>
<thead>
<tr>
<th>#</th>
<th>Class</th>
<th>No. of HTML documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Astronomy</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>Chemistry</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>Computer Science</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>Modern History</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>Ancient History</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>Physics</td>
<td>60</td>
</tr>
<tr>
<td>7</td>
<td>Sports</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>Medicine</td>
<td>45</td>
</tr>
<tr>
<td>9</td>
<td>Mathematics</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>420</td>
</tr>
</tbody>
</table>

### 4.3 Test Material

As query samples, three documents are taken from each of the nine classes, so as a total number of 27 documents is used as a query samples. The test aims to measure the precision, recall and the time required to execute each query. The whole test results of the control parameters are listed in Appendix (B).

In order to explore the effectiveness of each parameter on the system performance, the values of all parameters were set fixed except the value of the tested control parameter; its values was changed till
reaching its suitable value that leads to better overall precision, recall and time parameters.

4.4 Effect of Bonus Increment Parameter (α)

The role of this parameter is to control the degree of bonus given due to the existence of some styling attributes of the common keywords found in both of the database document and the queried document. Table (4.2) shows samples of the test results indicating that the best for bonus increment parameter is (0.1). So, this value is adopted in the coming tests.

<table>
<thead>
<tr>
<th>Query URL No.</th>
<th>Class</th>
<th>α=0</th>
<th>α=0.1</th>
<th>α=0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Time (s)</td>
<td>Prec.</td>
</tr>
<tr>
<td>1</td>
<td>Astronomy</td>
<td>0.83</td>
<td>0.62</td>
<td>6.47</td>
</tr>
<tr>
<td>2</td>
<td>0.80</td>
<td>0.60</td>
<td>28.8</td>
<td>0.97</td>
</tr>
<tr>
<td>3</td>
<td>0.80</td>
<td>0.60</td>
<td>25.5</td>
<td>0.97</td>
</tr>
<tr>
<td>4</td>
<td>Chemistry</td>
<td>0.50</td>
<td>0.38</td>
<td>3.45</td>
</tr>
<tr>
<td>5</td>
<td>0.83</td>
<td>0.62</td>
<td>6.22</td>
<td>0.97</td>
</tr>
<tr>
<td>6</td>
<td>0.93</td>
<td>0.70</td>
<td>3.92</td>
<td>1.00</td>
</tr>
<tr>
<td>7</td>
<td>CS</td>
<td>0.87</td>
<td>0.43</td>
<td>20.7</td>
</tr>
<tr>
<td>8</td>
<td>0.93</td>
<td>0.46</td>
<td>28.7</td>
<td>1.00</td>
</tr>
<tr>
<td>9</td>
<td>0.93</td>
<td>0.46</td>
<td>10.8</td>
<td>0.97</td>
</tr>
<tr>
<td>10</td>
<td>ModernH.</td>
<td>0.83</td>
<td>0.62</td>
<td>15.0</td>
</tr>
<tr>
<td>11</td>
<td>0.90</td>
<td>0.68</td>
<td>18.2</td>
<td>0.93</td>
</tr>
<tr>
<td>12</td>
<td>1.00</td>
<td>0.75</td>
<td>34.6</td>
<td>1.00</td>
</tr>
<tr>
<td>13</td>
<td>Math</td>
<td>0.83</td>
<td>0.41</td>
<td>13.2</td>
</tr>
<tr>
<td>14</td>
<td>0.90</td>
<td>0.60</td>
<td>6.19</td>
<td>0.93</td>
</tr>
<tr>
<td>15</td>
<td>0.87</td>
<td>0.57</td>
<td>8.06</td>
<td>0.93</td>
</tr>
<tr>
<td>16</td>
<td>Medicine</td>
<td>0.60</td>
<td>0.40</td>
<td>40.2</td>
</tr>
<tr>
<td>17</td>
<td>0.80</td>
<td>0.53</td>
<td>56.9</td>
<td>0.93</td>
</tr>
<tr>
<td>18</td>
<td>1.00</td>
<td>0.67</td>
<td>9.41</td>
<td>1.00</td>
</tr>
<tr>
<td>22</td>
<td>AncientH.</td>
<td>0.93</td>
<td>0.62</td>
<td>4.72</td>
</tr>
<tr>
<td>23</td>
<td>1.00</td>
<td>0.67</td>
<td>6.30</td>
<td>1.00</td>
</tr>
<tr>
<td>24</td>
<td>1.00</td>
<td>0.67</td>
<td>141.</td>
<td>1.00</td>
</tr>
<tr>
<td>25</td>
<td>Physics</td>
<td>0.93</td>
<td>0.47</td>
<td>9.64</td>
</tr>
<tr>
<td>26</td>
<td>0.80</td>
<td>0.40</td>
<td>6.70</td>
<td>0.83</td>
</tr>
<tr>
<td>27</td>
<td>0.86</td>
<td>0.43</td>
<td>24.7</td>
<td>0.90</td>
</tr>
<tr>
<td>28</td>
<td>Sport</td>
<td>0.70</td>
<td>0.47</td>
<td>140.</td>
</tr>
<tr>
<td>29</td>
<td>1.00</td>
<td>0.67</td>
<td>88.95</td>
<td>1.00</td>
</tr>
<tr>
<td>30</td>
<td>0.97</td>
<td>0.64</td>
<td>34.66</td>
<td>1.00</td>
</tr>
<tr>
<td>31</td>
<td>0.86</td>
<td>0.56</td>
<td>29.44</td>
<td>0.94</td>
</tr>
</tbody>
</table>
4.5 Effect of Matching Score Weight Factor (C)

Its role is to tune the effectiveness of the matching score parameter relative to the bonus parameter ($\alpha$). Table (4.3) illustrates the test results of three different values for this factor.

The test shows that choosing a number around 2.5 is the best, whereas, small numbers (less than 1) are not good and also large numbers (above 10) are not good as well. As shown in table (4.3) the best reached precision value is (0.97).

Table (4.3) Matching score weight factor

<table>
<thead>
<tr>
<th>Query URL No.</th>
<th>Class</th>
<th>$C=0.1$</th>
<th>$C=2.5$</th>
<th>$C=10$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Time (s)</td>
</tr>
<tr>
<td>1</td>
<td>Astronomy</td>
<td>0.83</td>
<td>0.6</td>
<td>4.16</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.93</td>
<td>0.70</td>
<td>30.98</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.93</td>
<td>0.70</td>
<td>33.03</td>
</tr>
<tr>
<td>4</td>
<td>Chemistry</td>
<td>0.93</td>
<td>0.6</td>
<td>3.28</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0.97</td>
<td>0.72</td>
<td>3.95</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.90</td>
<td>0.45</td>
<td>20.73</td>
</tr>
<tr>
<td>7</td>
<td>CS</td>
<td>0.97</td>
<td>0.48</td>
<td>30.22</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0.90</td>
<td>0.45</td>
<td>11.33</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>0.93</td>
<td>0.62</td>
<td>15.14</td>
</tr>
<tr>
<td>10</td>
<td>ModernH.</td>
<td>0.93</td>
<td>0.42</td>
<td>18.59</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>0.97</td>
<td>0.62</td>
<td>34.83</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>0.8</td>
<td>0.53</td>
<td>13.69</td>
</tr>
<tr>
<td>13</td>
<td>Math</td>
<td>0.8</td>
<td>0.53</td>
<td>6.02</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>0.8</td>
<td>0.53</td>
<td>13.69</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>0.93</td>
<td>0.62</td>
<td>34.83</td>
</tr>
<tr>
<td>16</td>
<td>Medicine</td>
<td>0.67</td>
<td>0.57</td>
<td>39.63</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>0.93</td>
<td>0.62</td>
<td>57.08</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>0.93</td>
<td>0.62</td>
<td>9.50</td>
</tr>
<tr>
<td>22</td>
<td>AncientH.</td>
<td>0.8</td>
<td>0.53</td>
<td>4.13</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>0.77</td>
<td>0.50</td>
<td>6.70</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>0.8</td>
<td>0.53</td>
<td>141.03</td>
</tr>
<tr>
<td>25</td>
<td>Physics</td>
<td>0.90</td>
<td>0.45</td>
<td>6.45</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>0.90</td>
<td>0.45</td>
<td>6.98</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>0.97</td>
<td>0.58</td>
<td>14.19</td>
</tr>
<tr>
<td>28</td>
<td>Soccer</td>
<td>0.8</td>
<td>0.60</td>
<td>144.63</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>0.93</td>
<td>0.70</td>
<td>90.17</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>0.97</td>
<td>0.72</td>
<td>38.09</td>
</tr>
</tbody>
</table>

| $DB_{\text{recsize}}=90, \alpha=0.1, Q_{\text{recsize}}=40, R=0.1$ |

<table>
<thead>
<tr>
<th>Query URL No.</th>
<th>Class</th>
<th>$C=0.1$</th>
<th>$C=2.5$</th>
<th>$C=10$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Time (s)</td>
</tr>
<tr>
<td>1</td>
<td>Astronomy</td>
<td>0.877</td>
<td>0.57</td>
<td>29.78</td>
</tr>
</tbody>
</table>
4.6 Effect of Database Document Record Size ($DB_{\text{recsize}}$)

This parameter represents the number of keywords taken to be matched from the tested database record. The test results show that taking a large number of keywords makes the retrieved information inaccurate (less precision). On the other hand, taking small number of keywords is not good as well, and the results will be irrational. The tests indicate that taking 90 keywords is convenient. Table (4.4) illustrates that the use of 90 keywords will lead to precision average of 0.97.

Table (4.4) Database document list size effect on the retrieval of TBIR

<table>
<thead>
<tr>
<th>Query URL No.</th>
<th>Class</th>
<th>$DB_{\text{recsize}=70}$</th>
<th></th>
<th>$DB_{\text{recsize}=90}$</th>
<th></th>
<th>$DB_{\text{recsize}=120}$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Time (s)</td>
<td>Prec.</td>
<td>Recall</td>
<td>Time (s)</td>
</tr>
<tr>
<td>1</td>
<td>Astronomy</td>
<td>0.97</td>
<td>0.725</td>
<td>6.47</td>
<td>1.00</td>
<td>0.75</td>
<td>4.20</td>
</tr>
<tr>
<td>2</td>
<td>Astronomy</td>
<td>0.97</td>
<td>0.72</td>
<td>28.84</td>
<td>0.97</td>
<td>0.72</td>
<td>40.78</td>
</tr>
<tr>
<td>3</td>
<td>Chemistry</td>
<td>0.97</td>
<td>0.725</td>
<td>25.58</td>
<td>1.00</td>
<td>0.75</td>
<td>38.73</td>
</tr>
<tr>
<td>4</td>
<td>Chemistry</td>
<td>0.83</td>
<td>0.62</td>
<td>3.45</td>
<td>0.93</td>
<td>0.70</td>
<td>3.19</td>
</tr>
<tr>
<td>5</td>
<td>Chemistry</td>
<td>0.97</td>
<td>0.72</td>
<td>6.22</td>
<td>0.97</td>
<td>0.72</td>
<td>6.19</td>
</tr>
<tr>
<td>6</td>
<td>CS</td>
<td>1.00</td>
<td>0.75</td>
<td>3.92</td>
<td>1.00</td>
<td>0.75</td>
<td>4.17</td>
</tr>
<tr>
<td>7</td>
<td>CS</td>
<td>0.97</td>
<td>0.48</td>
<td>20.75</td>
<td>1.00</td>
<td>0.50</td>
<td>24.78</td>
</tr>
<tr>
<td>8</td>
<td>CS</td>
<td>1.00</td>
<td>0.50</td>
<td>28.75</td>
<td>1.00</td>
<td>0.50</td>
<td>36.28</td>
</tr>
<tr>
<td>9</td>
<td>CS</td>
<td>0.97</td>
<td>0.48</td>
<td>10.83</td>
<td>0.97</td>
<td>0.48</td>
<td>10.94</td>
</tr>
<tr>
<td>10</td>
<td>CS</td>
<td>0.93</td>
<td>0.70</td>
<td>15.03</td>
<td>0.97</td>
<td>0.72</td>
<td>15.36</td>
</tr>
<tr>
<td>11</td>
<td>CS</td>
<td>0.93</td>
<td>0.70</td>
<td>18.23</td>
<td>0.93</td>
<td>0.70</td>
<td>18.58</td>
</tr>
<tr>
<td>12</td>
<td>CS</td>
<td>1.00</td>
<td>0.75</td>
<td>34.64</td>
<td>1.00</td>
<td>0.75</td>
<td>41.67</td>
</tr>
<tr>
<td>13</td>
<td>Math</td>
<td>0.86</td>
<td>0.43</td>
<td>13.20</td>
<td>0.93</td>
<td>0.62</td>
<td>13.34</td>
</tr>
<tr>
<td>14</td>
<td>Math</td>
<td>0.93</td>
<td>0.62</td>
<td>6.19</td>
<td>0.97</td>
<td>0.72</td>
<td>6.28</td>
</tr>
<tr>
<td>15</td>
<td>Math</td>
<td>0.93</td>
<td>0.62</td>
<td>8.06</td>
<td>0.93</td>
<td>0.62</td>
<td>7.92</td>
</tr>
<tr>
<td>16</td>
<td>Medicine</td>
<td>0.70</td>
<td>0.47</td>
<td>40.25</td>
<td>0.97</td>
<td>0.64</td>
<td>50.91</td>
</tr>
<tr>
<td>17</td>
<td>Medicine</td>
<td>0.93</td>
<td>0.62</td>
<td>56.95</td>
<td>0.93</td>
<td>0.62</td>
<td>69.52</td>
</tr>
<tr>
<td>18</td>
<td>Medicine</td>
<td>1.00</td>
<td>0.67</td>
<td>9.41</td>
<td>1.00</td>
<td>0.67</td>
<td>9.42</td>
</tr>
<tr>
<td>22</td>
<td>AncientH.</td>
<td>0.97</td>
<td>0.64</td>
<td>4.72</td>
<td>0.97</td>
<td>0.64</td>
<td>4.58</td>
</tr>
<tr>
<td>23</td>
<td>AncientH.</td>
<td>1.00</td>
<td>0.67</td>
<td>6.30</td>
<td>1.00</td>
<td>0.67</td>
<td>6.31</td>
</tr>
<tr>
<td>24</td>
<td>AncientH.</td>
<td>1.00</td>
<td>0.67</td>
<td>141.56</td>
<td>1.00</td>
<td>0.67</td>
<td>152.28</td>
</tr>
<tr>
<td>25</td>
<td>Physics</td>
<td>0.97</td>
<td>0.48</td>
<td>9.64</td>
<td>0.97</td>
<td>0.48</td>
<td>7.19</td>
</tr>
<tr>
<td>26</td>
<td>Physics</td>
<td>0.83</td>
<td>0.41</td>
<td>6.70</td>
<td>0.90</td>
<td>0.45</td>
<td>6.66</td>
</tr>
<tr>
<td>27</td>
<td>Physics</td>
<td>0.90</td>
<td>0.45</td>
<td>24.73</td>
<td>0.97</td>
<td>0.48</td>
<td>15.88</td>
</tr>
<tr>
<td>28</td>
<td>Soccer</td>
<td>0.73</td>
<td>0.48</td>
<td>140.73</td>
<td>0.87</td>
<td>0.48</td>
<td>148.67</td>
</tr>
<tr>
<td>29</td>
<td>Soccer</td>
<td>1.00</td>
<td>0.67</td>
<td>88.95</td>
<td>1.00</td>
<td>0.67</td>
<td>97.11</td>
</tr>
<tr>
<td>30</td>
<td>Soccer</td>
<td>1.00</td>
<td>0.67</td>
<td>34.66</td>
<td>1.00</td>
<td>0.67</td>
<td>42.05</td>
</tr>
</tbody>
</table>

|               |           | 0.94  | 0.61   | 29.44    | 0.97  | 0.63   | 32.70    | 0.90  | 0.59   | 33.31    |
4.7 Effect of Query Document Vector Window Size

A similar strategy test like that is followed in testing the effect of the database records size; the size of the query records is investigated as shown in table (4.5). The best size of the query document vector (\(Q_{\text{recsize}}\)) value is found to be 40 keywords.

As shown in the last raw of table (4.5) the best attained average precision value is 0.98. And also the time is increased by increasing the keywords.
4.8 Effect of S_ Function Exponent Factor (R)

The effect of this parameter is to tune the effectiveness of the weight parameter when applying the nonlinear mapping function (i.e., Hyperbolic Tangent $\tanh(x)$). This mapping function will soften the behavior of matching conditions against the highly weighted keywords. Table (4.6) illustrates that the best value of this parameter is 0.001 it leads to average precision equal 0.99.

Table (4.6) The exponent of the nonlinear function test

<table>
<thead>
<tr>
<th>Query URL No.</th>
<th>Class</th>
<th>DB$<em>{\text{recsize} = 90,\alpha=0.1, Q</em>{\text{recsize}}=40, C=2.5}$</th>
<th>R=0.1</th>
<th>Recall</th>
<th>Time (s)</th>
<th>R=0.001</th>
<th>Recall</th>
<th>Time (s)</th>
<th>R=0.00001</th>
<th>Prec.</th>
<th>Recall</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Astronomy</td>
<td>1.00 0.75</td>
<td>4.16</td>
<td>1 0.75</td>
<td>4.20</td>
<td>0.97</td>
<td>0.72</td>
<td>6.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.97 0.72</td>
<td>30.98</td>
<td>1 0.75</td>
<td>40.78</td>
<td>0.97</td>
<td>0.72</td>
<td>28.84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1.00 0.75</td>
<td>33.03</td>
<td>1 0.75</td>
<td>38.73</td>
<td>0.97</td>
<td>0.72</td>
<td>25.58</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Chemistry</td>
<td>0.97 0.72</td>
<td>3.28</td>
<td>1 0.75</td>
<td>3.19</td>
<td>0.93</td>
<td>0.7</td>
<td>3.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>0.97 0.72</td>
<td>5.88</td>
<td>0.97 0.72</td>
<td>6.19</td>
<td>0.97</td>
<td>0.72</td>
<td>6.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>1.00 0.75</td>
<td>3.95</td>
<td>1 0.75</td>
<td>4.17</td>
<td>0.97</td>
<td>0.72</td>
<td>3.92</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CS</td>
<td>1.00 0.50</td>
<td>20.73</td>
<td>1 0.5</td>
<td>24.78</td>
<td>0.97</td>
<td>0.58</td>
<td>20.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>1.00 0.50</td>
<td>30.22</td>
<td>1 0.5</td>
<td>36.28</td>
<td>0.9</td>
<td>0.45</td>
<td>28.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>CS</td>
<td>1.00 0.50</td>
<td>11.33</td>
<td>1 0.5</td>
<td>10.94</td>
<td>0.97</td>
<td>0.58</td>
<td>10.83</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>ModernH.</td>
<td>0.97 0.72</td>
<td>15.14</td>
<td>0.97 0.72</td>
<td>15.36</td>
<td>0.93</td>
<td>0.7</td>
<td>15.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>0.93 0.70</td>
<td>18.59</td>
<td>0.97 0.72</td>
<td>18.58</td>
<td>0.93</td>
<td>0.7</td>
<td>18.23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>1.00 0.75</td>
<td>34.83</td>
<td>1 0.75</td>
<td>41.67</td>
<td>1</td>
<td>0.75</td>
<td>34.64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Math</td>
<td>0.97 0.72</td>
<td>13.69</td>
<td>0.97 0.72</td>
<td>13.34</td>
<td>0.86</td>
<td>0.57</td>
<td>13.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>0.97 0.72</td>
<td>6.28</td>
<td>1 0.75</td>
<td>6.02</td>
<td>0.97</td>
<td>0.72</td>
<td>6.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>0.97 0.72</td>
<td>7.92</td>
<td>0.97 0.72</td>
<td>13.69</td>
<td>0.97</td>
<td>0.72</td>
<td>8.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Medicine</td>
<td>0.97 0.64</td>
<td>50.91</td>
<td>0.97 0.64</td>
<td>39.63</td>
<td>0.93</td>
<td>0.62</td>
<td>61.41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>0.93 0.62</td>
<td>69.52</td>
<td>0.97 0.64</td>
<td>57.08</td>
<td>0.93</td>
<td>0.62</td>
<td>67.36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>1.00 0.67</td>
<td>9.42</td>
<td>1 0.67</td>
<td>9.50</td>
<td>0.97</td>
<td>0.64</td>
<td>9.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>AncientH.</td>
<td>0.97 0.64</td>
<td>8.86</td>
<td>0.97 0.64</td>
<td>9.22</td>
<td>0.93</td>
<td>0.62</td>
<td>9.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>1.00 0.67</td>
<td>45.98</td>
<td>1 0.67</td>
<td>66.95</td>
<td>0.97</td>
<td>0.64</td>
<td>61.13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>1.00 0.67</td>
<td>5.39</td>
<td>1 0.67</td>
<td>5.64</td>
<td>0.97</td>
<td>0.64</td>
<td>5.84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Physics</td>
<td>0.97 0.58</td>
<td>4.58</td>
<td>0.97 0.48</td>
<td>4.13</td>
<td>0.9</td>
<td>0.45</td>
<td>5.11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>0.90 0.45</td>
<td>6.31</td>
<td>0.9 0.53</td>
<td>6.70</td>
<td>0.9</td>
<td>0.45</td>
<td>7.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>0.97 0.58</td>
<td>152.28</td>
<td>0.97 0.48</td>
<td>141.03</td>
<td>0.83</td>
<td>0.41</td>
<td>150.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Soccer</td>
<td>0.97 0.64</td>
<td>7.19</td>
<td>1 0.67</td>
<td>6.98</td>
<td>0.87</td>
<td>0.55</td>
<td>6.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>1.00 0.67</td>
<td>15.88</td>
<td>1 0.67</td>
<td>14.19</td>
<td>0.97</td>
<td>0.64</td>
<td>20.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>0.98 0.66</td>
<td>23.07</td>
<td>0.99 0.66</td>
<td>23.90</td>
<td>0.941</td>
<td>0.63</td>
<td>23.82</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.9 Text Operations Effect

The stemming, removal of the stop words and text cleansing operations have a notable effect on the retrieved documents. In the following subsection the effects of these operations are clarified.

4.9.1 Stemming/Stop Word Removal Dependency

As stated in table (4.7) removal of the stop words will be increased after stemming about 56%. The first column in table (4.7) is the query number and the second is the percentage of character reduction in case of no stemming operation, while the third column is the reduction percentage when stemming operation is conducted.

<table>
<thead>
<tr>
<th>Q no.</th>
<th>without stemming</th>
<th>with stem</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.23%</td>
<td>66.65%</td>
</tr>
<tr>
<td>2</td>
<td>20.69%</td>
<td>76.69%</td>
</tr>
<tr>
<td>3</td>
<td>20.34%</td>
<td>64.72%</td>
</tr>
<tr>
<td>4</td>
<td>17.67%</td>
<td>37.13%</td>
</tr>
<tr>
<td>5</td>
<td>29.08%</td>
<td>44.43%</td>
</tr>
<tr>
<td>6</td>
<td>24.87%</td>
<td>38.89%</td>
</tr>
<tr>
<td>7</td>
<td>14.11%</td>
<td>67.26%</td>
</tr>
<tr>
<td>8</td>
<td>14.47%</td>
<td>62.43%</td>
</tr>
<tr>
<td>9</td>
<td>24.29%</td>
<td>49.72%</td>
</tr>
<tr>
<td>10</td>
<td>27.17%</td>
<td>42.06%</td>
</tr>
<tr>
<td>11</td>
<td>4.58%</td>
<td>86.63%</td>
</tr>
<tr>
<td>12</td>
<td>9.93%</td>
<td>70.58%</td>
</tr>
<tr>
<td>13</td>
<td>27.56%</td>
<td>55.97%</td>
</tr>
<tr>
<td>14</td>
<td>18.35%</td>
<td>53.49%</td>
</tr>
<tr>
<td>15</td>
<td>19.24%</td>
<td>54.66%</td>
</tr>
<tr>
<td>16</td>
<td>22.28%</td>
<td>48.39%</td>
</tr>
<tr>
<td>17</td>
<td>17.06%</td>
<td>48.78%</td>
</tr>
<tr>
<td>18</td>
<td>20.35%</td>
<td>41.92%</td>
</tr>
<tr>
<td>19</td>
<td>21.27%</td>
<td>52.04%</td>
</tr>
<tr>
<td>20</td>
<td>19.79%</td>
<td>49.63%</td>
</tr>
<tr>
<td>21</td>
<td>24.55%</td>
<td>42.50%</td>
</tr>
<tr>
<td>22</td>
<td>10.62%</td>
<td>60.88%</td>
</tr>
<tr>
<td>23</td>
<td>8.75%</td>
<td>73.92%</td>
</tr>
<tr>
<td>24</td>
<td>21.19%</td>
<td>50.94%</td>
</tr>
<tr>
<td>25</td>
<td>25.68%</td>
<td>40.99%</td>
</tr>
<tr>
<td>26</td>
<td>25.68%</td>
<td>40.99%</td>
</tr>
<tr>
<td>27</td>
<td>20.21%</td>
<td>57.48%</td>
</tr>
<tr>
<td>28</td>
<td>18.38%</td>
<td>59.25%</td>
</tr>
<tr>
<td>29</td>
<td>17.40%</td>
<td>59.82%</td>
</tr>
<tr>
<td>30</td>
<td>17.40%</td>
<td>59.82%</td>
</tr>
<tr>
<td>31</td>
<td>17.67%</td>
<td>76.07%</td>
</tr>
<tr>
<td>32</td>
<td>27.19%</td>
<td>70.22%</td>
</tr>
<tr>
<td>33</td>
<td>18.95%</td>
<td>75.06%</td>
</tr>
<tr>
<td>34</td>
<td>40.19%</td>
<td>60.56%</td>
</tr>
<tr>
<td>35</td>
<td>27.22%</td>
<td>47.85%</td>
</tr>
<tr>
<td>36</td>
<td>19.28%</td>
<td>43.91%</td>
</tr>
<tr>
<td>37</td>
<td>16.04%</td>
<td>50.96%</td>
</tr>
<tr>
<td>38</td>
<td>18.85%</td>
<td>44.79%</td>
</tr>
<tr>
<td>39</td>
<td>17.39%</td>
<td>64.08%</td>
</tr>
<tr>
<td>40</td>
<td>20.93%</td>
<td>40.45%</td>
</tr>
<tr>
<td>41</td>
<td>16.22%</td>
<td>63.38%</td>
</tr>
</tbody>
</table>

Table (4.7) Stop words removal and stemming relationship
4.9.2 Text Operations Effect on Retrieval

The main text operations are stemming and stop word detection and removal; and in order to evaluate their effectiveness on retrieval then, system performance measures and the time is calculated with and without applying these text operations. The tests show that applying these operations has increased the precision about 9.5% and the recall about 10.7%, but, the execution time has increased about 45%, as shown in table (4.8).

4.10 Nonlinear Function Effect on the Retrieval

This function is utilized as a membership function to map the score of overall matching results for the query record and a database record. The effect of using this function (i.e., \( \tanh(x) \)) is investigated in terms of precision, recall and time, as shown in table (4.8). The test shows that using s-mapping function has enhanced the system accuracy performance.

<table>
<thead>
<tr>
<th>Query URL No.</th>
<th>Class</th>
<th>WITH TEXT OP.</th>
<th>WITHOUT TEXT OP.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
</tr>
<tr>
<td>1</td>
<td>Astronomy</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>2</td>
<td>Astronomy</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>3</td>
<td>Astronomy</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>Chemistry</td>
<td>1</td>
<td>0.66</td>
</tr>
<tr>
<td>5</td>
<td>Chemistry</td>
<td>0.8</td>
<td>0.53</td>
</tr>
<tr>
<td>6</td>
<td>Chemistry</td>
<td>1</td>
<td>0.66</td>
</tr>
<tr>
<td>7</td>
<td>CS</td>
<td>0.93</td>
<td>0.48</td>
</tr>
<tr>
<td>8</td>
<td>CS</td>
<td>0.7</td>
<td>0.36</td>
</tr>
<tr>
<td>9</td>
<td>CS</td>
<td>0.93</td>
<td>0.48</td>
</tr>
<tr>
<td>10</td>
<td>Modern History</td>
<td>0.97</td>
<td>0.72</td>
</tr>
<tr>
<td>11</td>
<td>Modern History</td>
<td>0.9</td>
<td>0.68</td>
</tr>
<tr>
<td>12</td>
<td>Modern History</td>
<td>1</td>
<td>0.67</td>
</tr>
</tbody>
</table>
### Chapter Four

#### Test

<table>
<thead>
<tr>
<th>Query URL No.</th>
<th>Class</th>
<th>WITH TEXT OP.</th>
<th>WITHOUT TEXT OP.</th>
<th>Prec.</th>
<th>Recall</th>
<th>Time (s)</th>
<th>Prec.</th>
<th>Recall</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Math</td>
<td>0.83</td>
<td>0.53</td>
<td>5.5</td>
<td>0.7</td>
<td>0.47</td>
<td>3.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>0.73</td>
<td>0.49</td>
<td>10.45</td>
<td>0.5</td>
<td>0.33</td>
<td>6.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>0.97</td>
<td>0.65</td>
<td>13.39</td>
<td>0.9</td>
<td>0.6</td>
<td>7.78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Medicine</td>
<td>1</td>
<td>0.75</td>
<td>16.13</td>
<td>0.93</td>
<td>0.62</td>
<td>9.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Medicine</td>
<td>1</td>
<td>0.75</td>
<td>61.11</td>
<td>1</td>
<td>0.75</td>
<td>41.73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>1</td>
<td>0.75</td>
<td>17.09</td>
<td>0.97</td>
<td>0.72</td>
<td>9.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>1</td>
<td>0.66</td>
<td>3.13</td>
<td>0.63</td>
<td>0.42</td>
<td>2.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>AncientH.</td>
<td>0.87</td>
<td>0.57</td>
<td>13.94</td>
<td>0.77</td>
<td>0.51</td>
<td>7.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>0.9</td>
<td>0.6</td>
<td>2.59</td>
<td>0.83</td>
<td>0.56</td>
<td>2.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Physics</td>
<td>1</td>
<td>0.5</td>
<td>87.75</td>
<td>0.93</td>
<td>0.57</td>
<td>59.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Physics</td>
<td>0.93</td>
<td>0.57</td>
<td>11.09</td>
<td>0.93</td>
<td>0.57</td>
<td>7.19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>1</td>
<td>0.5</td>
<td>18.8</td>
<td>1</td>
<td>0.5</td>
<td>11.97</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Soccer</td>
<td>1</td>
<td>0.67</td>
<td>40.02</td>
<td>0.93</td>
<td>0.51</td>
<td>27.59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Soccer</td>
<td>1</td>
<td>0.67</td>
<td>34.08</td>
<td>0.97</td>
<td>0.64</td>
<td>24.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Soccer</td>
<td>1</td>
<td>0.67</td>
<td>57.63</td>
<td>1</td>
<td>0.67</td>
<td>39.61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean Value</td>
<td></td>
<td>0.94</td>
<td>0.62</td>
<td>29.81</td>
<td>0.86</td>
<td>0.56</td>
<td>20.81</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fig. (4.1) The effect of text operation on the retrieval process**
### Table (2.9) The effect of s-mapping function

<table>
<thead>
<tr>
<th>Query URL No.</th>
<th>Class</th>
<th>WITH FUZZY</th>
<th></th>
<th>WITHOUT FUZZY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Time (s)</td>
<td>Prec.</td>
</tr>
<tr>
<td>1</td>
<td>Astronomy</td>
<td>1</td>
<td>0.75</td>
<td>21.36</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>Chemistry</td>
<td>1</td>
<td>0.75</td>
<td>92.25</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>CS</td>
<td>0.93</td>
<td>0.48</td>
<td>7.36</td>
<td>0.8</td>
</tr>
<tr>
<td>4</td>
<td>ModernH.</td>
<td>0.97</td>
<td>0.72</td>
<td>32.28</td>
<td>0.73</td>
</tr>
<tr>
<td>5</td>
<td>Math</td>
<td>0.97</td>
<td>0.65</td>
<td>13.39</td>
<td>0.93</td>
</tr>
<tr>
<td>6</td>
<td>Medicine</td>
<td>0.83</td>
<td>0.53</td>
<td>5.50</td>
<td>0.93</td>
</tr>
<tr>
<td>7</td>
<td>AncientH.</td>
<td>0.87</td>
<td>0.57</td>
<td>13.94</td>
<td>0.87</td>
</tr>
<tr>
<td>8</td>
<td>Physics</td>
<td>0.93</td>
<td>0.62</td>
<td>2.73</td>
<td>0.9</td>
</tr>
<tr>
<td>9</td>
<td>Soccer</td>
<td>1</td>
<td>0.67</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>3.13</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>3.13</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td>0.97</td>
<td>0.65</td>
</tr>
<tr>
<td>13</td>
<td>Medicine</td>
<td>0.83</td>
<td>0.53</td>
<td>5.50</td>
<td>0.93</td>
</tr>
<tr>
<td>14</td>
<td>Math</td>
<td>0.73</td>
<td>0.49</td>
<td>10.45</td>
<td>0.86</td>
</tr>
<tr>
<td>15</td>
<td>Medicine</td>
<td>1</td>
<td>0.75</td>
<td>16.13</td>
<td>0.97</td>
</tr>
<tr>
<td>16</td>
<td>AncientH.</td>
<td>0.87</td>
<td>0.57</td>
<td>13.94</td>
<td>0.87</td>
</tr>
<tr>
<td>17</td>
<td>Soccer</td>
<td>1</td>
<td>0.67</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td>0.97</td>
<td>0.65</td>
</tr>
<tr>
<td>19</td>
<td>Physics</td>
<td>0.93</td>
<td>0.62</td>
<td>2.73</td>
<td>0.9</td>
</tr>
<tr>
<td>20</td>
<td>ModernH.</td>
<td>1</td>
<td>0.66</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>Soccer</td>
<td>1</td>
<td>0.67</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td>0.97</td>
<td>0.65</td>
</tr>
<tr>
<td>23</td>
<td>Physics</td>
<td>0.93</td>
<td>0.62</td>
<td>2.73</td>
<td>0.9</td>
</tr>
<tr>
<td>24</td>
<td>AncientH.</td>
<td>0.87</td>
<td>0.57</td>
<td>13.94</td>
<td>0.87</td>
</tr>
<tr>
<td>25</td>
<td>Soccer</td>
<td>1</td>
<td>0.67</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td>0.97</td>
<td>0.65</td>
</tr>
<tr>
<td>27</td>
<td>Physics</td>
<td>0.93</td>
<td>0.62</td>
<td>2.73</td>
<td>0.9</td>
</tr>
<tr>
<td>28</td>
<td>ModernH.</td>
<td>1</td>
<td>0.66</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>Soccer</td>
<td>1</td>
<td>0.67</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td>0.97</td>
<td>0.65</td>
</tr>
</tbody>
</table>

It is clear from the table that the precision is increased about 5.6%, while the recall increased about 2.63% and the execution time increased about 17.3%.

### 4.11 Frequency-Based Search vs. exhaustive Search

The method used to match two records is well-explained in chapter 3, in this chapter the experiments are performed over some queries samples, firstly by applying the brute force as a search method which impies trial of all choices, and then secondly the same set of sample queries are tested using the proposed and applied method.
This test is performed to assess the efficacy of the proposed method, as illustrated in table (2.10). The results show that applying this method has fastened the retrieval about 16.6%.

**Table (2.10) The Frequency-based matching and search vs. the traditional Executive search**

<table>
<thead>
<tr>
<th>Frequency-based search</th>
<th>Executive Search</th>
<th>Frequency-based search</th>
<th>Executive Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.92</td>
<td>15.33</td>
<td>15.27</td>
<td>15.66</td>
</tr>
<tr>
<td>14.08</td>
<td>15.23</td>
<td>7.27</td>
<td>7.48</td>
</tr>
<tr>
<td>29.52</td>
<td>35.64</td>
<td>45.56</td>
<td>50.11</td>
</tr>
<tr>
<td>58.67</td>
<td>67.34</td>
<td>9.45</td>
<td>10.81</td>
</tr>
<tr>
<td>63.75</td>
<td>69.05</td>
<td>48.00</td>
<td>56.14</td>
</tr>
<tr>
<td>18.78</td>
<td>25.73</td>
<td>17.50</td>
<td>15.59</td>
</tr>
<tr>
<td>35.52</td>
<td>42.98</td>
<td>20.17</td>
<td>28.09</td>
</tr>
<tr>
<td>9.17</td>
<td>9.52</td>
<td>22.91</td>
<td>44.98</td>
</tr>
<tr>
<td>16.50</td>
<td>16.27</td>
<td>6.75</td>
<td>7.13</td>
</tr>
<tr>
<td>37.45</td>
<td>52.78</td>
<td>50.61</td>
<td>58.89</td>
</tr>
<tr>
<td>101.66</td>
<td>109.08</td>
<td>9.86</td>
<td>9.06</td>
</tr>
<tr>
<td>17.28</td>
<td>25.67</td>
<td>13.25</td>
<td>19.45</td>
</tr>
<tr>
<td>19.64</td>
<td>26.08</td>
<td>7.50</td>
<td>7.80</td>
</tr>
<tr>
<td>40.55</td>
<td>49.91</td>
<td>10.06</td>
<td>10.69</td>
</tr>
<tr>
<td>18.66</td>
<td>19.20</td>
<td>10.66</td>
<td>11.06</td>
</tr>
<tr>
<td>7.67</td>
<td>7.91</td>
<td>29.75</td>
<td>36.36</td>
</tr>
<tr>
<td>44.63</td>
<td>51.47</td>
<td>24.30</td>
<td>24.39</td>
</tr>
<tr>
<td>38.31</td>
<td>66.69</td>
<td>51.47</td>
<td>58.17</td>
</tr>
</tbody>
</table>

**Average Time**

<table>
<thead>
<tr>
<th>Frequency-based search</th>
<th>Executive Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>27.42</td>
<td>32.72</td>
</tr>
</tbody>
</table>
Chapter Five
Conclusions and Suggestions

5.1 Conclusions
1. Using text operations helped to increase precision by 9.5% and recall by 10.7%, while the execution time is increased about 45%.
2. The use of nonlinear s-mapping membership function increased the precision by about 5.6%, and recall by 2.63%, while the time is increased by about 17.3%.
3. The system retrieves the documents intelligently and automatically without user intervention and without assessment whether a keyword is important, the system will automatically determine the significance of words, and then retrieves the documents according the significant keywords that the user didn’t input to the system. This infers that the system has 0 input.
4. After tuning the system control parameters, as illustrated in chapter 4 the best attained precision and recall become 0.99 and 0.66, respectively.
5. After stemming, the reduction percentage of characters of stop words is reduced to 56%.
6. Using the Frequency-based search the matching method has fastened the retrieval by about 16.6%, in comparison with using the traditional Exhaustive search method.

5.2 Suggestions for Future Work
1. Since the matching process is involved with many-to-many matches, so, there is a need to develop an indexing system that can handle efficiently the matching process on a selective bases (for example using a more sophisticated hash function.)
2. The adopted stemming algorithm used can be substituted by another one. But the proposed stemming has achieved relatively rational results.

3. The system retrieves only English documents, it can be developed to be multilingual system, being able to retrieve documents in any language desired to be retrieved.

4. Another kind of information containers could be used instead of the HTML (e.g. XML, PDF, and so on).

5. The architecture of the system is based on client/server model. So, the system is ready to be put on a server computer which is connected to its clients through a network, sending a request to the server over greed-upon protocol, and then the server grants the request to clients, immediately.

6. Other soft computing mapping function may be used to enhance the retrieval with or without the fuzzy logic.

7. The structural codes are not considered at all in the system, since the TBIR is a content mining application. It can be extended to also retrieve the documents according to their structural similarity rather than depending on the text only. So, JAVASCRIPT and CSS codes can be used.

8. The set of words feature styling tags can include another attributes, and also the META tag can be used.

Note: that the proposed system is not a prototype of a Search Engine it is an Information Retrieval System which acts in similar way to the user behavior.
References


Chapter 1

General Introduction
Chapter 2

Mining Based on Text Contents
Chapter 3

Design and Implementation Of TBIR
Chapter 4

Test Results
Chapter 5

Conclusions and Suggestions
For Future Work
Appendix A
Appendix B
الخلاصة

إن بيئة الإنترنت هي مصدر مهم وغني بالمعلومات ومتزايد باستمرار، فاصبح من الضروري استحداث أنظمة تعمل على اكتشاف و"التنقيب" ما هو مفيد من بيئة الإنترنت بشكل ذاكي (أي بدون تدخل المستخدم في عملية البحث) وبشكل سريع. هذا النوع من الأنظمة يساعد المستخدم في الحصول على المعلومات باقل جهد وقابل وقت من طرق البحث التقليدية والتي تعتمد على الشخص نفسه في البحث. أن هدف الدراسة هو بناء منظومة استرجاع الوثائق (HTML) وذلك بالإضافة إلى التماثل النصي لهذا الوثائق.

إن التماثل بالاعتماد على المحتوى النصي يتطلب تحليل النص وأجراء الإحصائيات أي تحويل النص إلى أرقام يتم التعامل معها رياضيا، فيتم تحويلها إلى منتج رياضي يمثل النص التتابع لتلك الوثيقة وإجراء المطابقة بالاعتماد على تلك المنتجات ويتم ذلك بتمرير النص على عمليات معينة تقوم بذلك الغرض ويقوم "التنقيب النصي" وتحويل النص إلى معلومات إحصائية مفيدة. 
بعد هذه المرحلة يتم التفويض وفقاً للنتائج هذه العمليات بين المنتجات. إن ما لوحظ هو أن الطرق التقليدية لا تساعد في التماثل بين تلك المنتجات لا تساعي النتائج الواقعية والمنطقية بالشكل المطلوب كاستخدام البحث الهندسي بين متجهين، فوجد أن التعامل مع تلك الإحصائيات في نوع من التعقيد مشكلة الحالية، في العمل المفترض تم التعامل معها لغرض تحقيق المطابقة على نموذج منطقية يعتمد أساساً على "المنطق الضبب" والذي يعني شيء من الواقعية للنتائج أكثر من الاعتماد على الطرق التقليدية.

ان النظام المتкурر يتالف من مراحلتين؛ الأولى هي مرحلة "التسجيل"، في هذه المرحلة يتم تخزين الوثائق (صفحات الHTML المخصصة للنظام) بالإضافة إلى المنتجات الممثلة لها في قاعدة البيانات. أما المرحلة الثانية فهي مرحلة "الاسترجاع" ويتم ذلك بعد إدخال عنوان صفحة الإنترنت فيقوم النظام بتحليل الصفحة المعنونة بهذا العنوان بنفس الطريقة التي تم فيها تحليل الوثائق بمرحلة التسجيل وتميرها على نفس العمليات وتحويلها إلى نفس الشكل الرياضي للوثائق المسجلة. وذللك يتم المطابقة بين متجه وثيقة المستخدم مع جميع المنتجات المخزونة. إن نتيجة المطابق تسمى "المرتبة"، وفقاً إلى المرتبة التي حصل عليها التناقض يتم "تصنيف الوثيقة"، فكلما ازدادت المرتبة ازداد التماثل بين الوثائقين في هذا الشكل يتم استرجاع الوثائق المطلوبة التي تكون مماثلة مع وثيقة المستخدم.

إذاً بعد خروج النظام عن مرحلة مراحل الإعداد "المقدمة" والاعتماد، وان أفضل النتائج التي حققاً النظام هي (0.66) للاستدعاء و (0.99) للدقة.
تصميم وتطبيق منظومة استرجاع للمعلومات النصية بالاعتماد على المنطق المضبب

رسالة
مقدمة إلى كلية العلوم في جامعة بغداد كجزء من متطلبات نيل درجة ماجستير علوم في علوم الحاسبات

من قبل
ابراهيم عامر حميد
(بكالوريوس 2008)

بباشراف
د. لوي ادودد جورج